Self-Similarity and Wavelet Transforms for the Compression of Still Image and Video Data

Ian Karl Levy B.Sc.

A thesis submitted to
The University of Warwick
for the degree of
Doctor of Philosophy

February 1998

Self-Similarity and Wavelet Transforms for the Compression of Still Image and Video Data

Ian Karl Levy B.Sc.

A thesis submitted to
The University of Warwick
for the degree of
Doctor of Philosophy
February 1998

This thesis is concerned with the methods used to reduce the data volume required to represent still images and video sequences. The number of disparate still image and video coding methods increases almost daily. Recently, two new strategies have emerged and have stimulated widespread research. These are the fractal method and the wavelet transform. In this thesis, it will be argued that the two methods share a common principle: that of *self-similarity*. The two will be related concretely via an image coding algorithm which combines the two, normally disparate, strategies.

The wavelet transform is an orientation selective transform. It will be shown that the selectivity of the conventional transform is not sufficient to allow exploitation of self-similarity while keeping computational cost low. To address this, a new wavelet transform is presented which allows for greater orientation selectivity, while maintaining the orthogonality and data volume of the conventional wavelet transform. Many designs for vector quantizers have been published recently and another is added to the gamut by this work. The tree structured vector quantizer presented here is on-line and self structuring, requiring no distinct training phase. Combining these into a still image data compression system produces results which are among the best that have been published to date.

An extension of the two dimensional wavelet transform to encompass the time dimension is straightforward and this work attempts to extrapolate some of its properties into three dimensions. The vector quantizer is then applied to three dimensional image data to produce a video coding system which, while not optimal, produces very encouraging results.

Key Words:
Wavelet Transform, Fractal Image Compression, Iterated Function Systems, Orientation, Image Compression, Video Compression, Multiresolution Analysis, Vector Quantization

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council and conducted within the Image and Signal Processing Research Group in the Department of Computer Science at the University of Warwick, UK.

Thanks are due to the members of the Image and Audio Signal Processing Group for their friendship and providing advice and often welcome diversions during my time with them, in particular: James Beacom, Nicola Cross, Andy King, Chang-Tsun Li, Peter Meulemans and Tim Shuttleworth.

Special mention must go to my supervisor, Roland Wilson, without whom this work would not have been possible. His ideas, support and, above all, encouragement have been truly appreciated.

Finally, I would like to thank Susan, my friends and my family for the support they have given me over the years.

Contents

List of Figures

List of Tables

List of Symbols

This table contains a summary of the mathematical notation used in this thesis. Note that only notation that carries meaning across a large part of the thesis is listed. Also note that local definitions may supersede those listed here.

Symbol	*Description*
E	An entropy
$\mathcal{E}$	Expected value
ϵ	An error
x_i	A signal
$\hat{x}_i$	An approximation to a signal
d	A distortion measure
T	A transform matrix
T^{-1}	An inverse transform matrix
$\Re$	The set of real numbers
s	A contractivity factor
$x^\star$	The fixed point of a contraction mapping
W	An IFS
D_j	A domain block
R_i	A range block
$\hat{R}_i$	A normalised range block
$\hat{D}_j$	An approximation to a domain block
ι_c	An isometry
X	A metric space
$e(x)$	An error function
k	A shift
α	A scale factor
ψ	A basis function
$\psi_{i,j}$	A scaled and shifted basis function
V_n	A vector space
C_x	A covariance matrix
h_n, g_n	A quadrature mirror filter pair
$\mathcal{N}$	A node in a TSVQ
$z_{\mathcal{N}}$	The decision hyperplane direction at node $\mathcal{N}$
$a_{\mathcal{N}}$	The decision hyperplane intersection at node $\mathcal{N}$
$d_{\mathcal{N}}$	The average distortion at node $\mathcal{N}$

Chapter 1

Introduction and Scope of Thesis

1.1 Background

The recent explosion of global communication systems and interconnectivity has resulted in massive growth of the number of connected computer users. These users are connected by a common set of protocols, running over the physical media that have come to be known as the *Internet.* This explosion in the number of users has, in turn, resulted in a dramatic increase in the amount of data passed over the Internet. The *World Wide Web (WWW)* is an embodiment of the *HTTP* protocol and provides a rich, multimedia experience to anyone equipped with a piece of browser software and an Internet connection. This multimedia experience consists of text, images, audio and indeed video. As with any physical medium, the Internet has bandwidth limitations. That is, the amount of data that can flow through any point on the Internet in a given time period is limited. Now, a single image frame at broadcast television quality in monochrome occupies approximately 300kB of computer storage, roughly the same as a two hundred and fifty page text document. Consequently, the shift towards multimedia communication and away from plain text increases the volume of data to be transmitted by several orders of magnitude. Couple this with the continual growth of the number of users and it is simple to see that the limited bandwidth of the Internet is a significant constraint on its usefulness. Methods of reducing the volume of data to be sent in all cases are required.

The advent of high definition digital television (HDTV), cable TV, digital broadcasting direct to homes and the possibility of "video-on-demand" over existing telecommunication systems further increases the need for image and video compression technologies. Given a standard Basic Rate Interface ISDN channel into a home, the 128 kilobits per second bandwidth is soon used up by a simple video stream alone. Consider an image that is approximately quarter TV-size (QCIF format [Cla95]). This would consist of 144×276 pixels, occupying 317952 bits. Given the $128kbps$ bandwidth, this would only allow three frames per second to be transmitted, hardly enough for the viewer to perceive any motion at all and certainly not even close to the quality produced by the current analogue television system.

1.2 Digital Signal Representation

If digital systems cannot inherently match the quality of analogue ones, why bother with them? In a digital transmission system, it is simple to differentiate between signal and noise, unlike in the analogue system. The data that are sent are *exactly* the data received. Thus, digital systems can operate over much larger distances by the use of simple repeaters. Digital representations of signals also allow computers and other digital processors to operate directly upon the data, without conversion and the associated losses. Indeed, digital representations are discrete and, as such, much simpler to operate upon than their analogue counterparts. The advent of digital signal representations has made tractable many problems that are intractable in the analogue domain. The applications made possible by the digital signal representation are too numerous to list. For example, in image analysis parameters are extracted which somehow describe the content and structure of an image in order to perform such tasks as object detection, recognition and classification. Image enhancement appears in many forms. For example, reducing the amount of noise in an image so that it is more pleasing to the human observer or false colouring images, such as satellite and medical images, to aid human analysis.

In this thesis, it is always assumed that digital images are stored as a Cartesian array of 8-bit integer values specifying the grey scale intensity at that point. The array may be of arbitrary size, but will usually be 256×256 or 512×512 elements

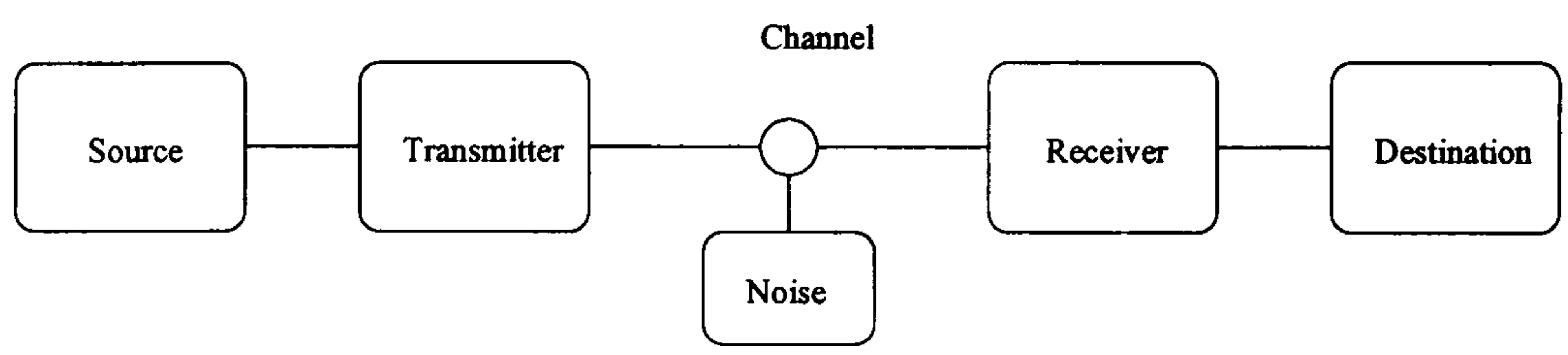


Figure 1.1: Shannon's model of communication

(pixels) for simplicity of processing.

1.3 Foundations

Data compression has existed, in one form or another, since the dawn of human communication. A trivial example is the abbreviation of phrases such as "do not" to "don't" and "will not" to "won't". A more substantial example is Morse code. In this system, more common letters are represented by shorter code words. For example, E is represented by the codeword $(\cdot)$ while the less common Y is $(- \cdot --)$.

The founding work for all information theory was that of Shannon [Sha48]. He proposed the communication model shown in figure 1.1. The source is a process that produces symbols, taken from a finite *source alphabet*, which can be taken to represent a given signal in digital form. The transmitter takes these symbols and performs some process on them. The output of the transmitter is a stream of symbols taken from the (possibly different) *channel alphabet*. The receiver takes the channel symbols and performs the inverse operation to the transmitter. The resultant signal is then passed to the destination.

It is prudent to further split the coding into *source coding* and *channel coding*. Consider a channel with a capacity, C symbols per second, and a source with entropy E symbols per second. Assuming that $E \leq C$, Shannon, in his *Fundamental Theorem for a Discrete Channel with Noise* [Sha48] stated that

> It is possible to send information at the rate C through the channel with as small a frequency of errors or equivocations as desired by proper encoding.

This ensures that, given the output of an arbitrary source satisfying the entropy constraint, it is possible to send this via the channel with arbitrary precision using a suitable channel code. Thus, it is possible and also convenient practically, to separate *source* and *channel* coding: the former aims at producing the most efficient representation of the source information, while the latter is aimed at minimising the errors caused by the channel. The source coding problem can be seen as an application of rate-distortion theory [Sha59] [Ber68] [Pin69]: given a fidelity criterion, the task is to encode the source output to produce the minimum amount of information. Conversely, given a maximum data rate, the task is to maximize any fidelity measure (i.e. minimize any distortion measure). The minimizing of the error and the amount of information to be transmitted is the goal of all data compression algorithms.

This general communication model covers many processes and, in particular, image data compression. The purpose of the system is to reconstruct an approximation of the source data at the destination. The destination will usually be a human observer and any distortion measures should reflect this fact. The channel may be a physical channel, for example radio or optical fibre, or simply disk storage. For the purposes of this thesis, the properties of the channel are moot since it is concerned only with source coding, not channel coding. Indeed, in the two examples cited, the respective channel codings ensure "near lossless" delivery to the destination.

The transmitter takes the source image and produces a stream of channel symbols. If there are fewer channel symbols than source symbols, compression has been achieved, assuming the cost of transmission is constant across all channel symbols. If the operation performed by the transmitter is invertible, then the compression is termed *lossless*, i.e. the reconstruction at the destination will be perfect. If the operation performed is not invertible, then the compression is termed *lossy* and a distortion measure is used to determine the quality of the reconstruction. Shannon [Sha48] also defined the *entropy* of a signal. This is the minimum number of bits required to encode a given signal. If the signal values are independent, then the entropy of the digitised signal is given by

$$E = -\sum_{i=1}^{N} P(i) \log_2(P(i)) \tag{1.1}$$

where N is the number of symbols in the alphabet and $P(i)$ is the probability of the symbol i occuring. For the *Lena* image shown in figure 1.2, this gives an entropy of 7.25 bits per pixel. However, image pixel values are rarely independent and it

Figure 1.2: Original Lena 512×512 pixel image

is often prudent to take the previous value into account. This is measured via the first order Shannon entropy, given by

$$E = -\sum_{j=1}^{N} P(j) \sum_{i=1}^{N} P(i|j) \log_2(P(i|j)) \tag{1.2}$$

where $P(i|j)$ is the probability that symbol i will occur given that symbol j has just occurred. The first order entropy of the *Lena* image in figure 1.2 is 4.65 bits per pixel. It is common to utilise the correlation between samples in a signal. By inspection, it is obvious that natural images have correlations that extend over relatively large distances. These would not be fully accounted for by the first order Shannon entropy, which only takes into account the immediately preceding sample. One of the aims of this thesis is to derive methods of exploiting these correlations, regardless of their distance or position.

As an example, the wavelet transform (see chapter 2) was applied to the *Lena*

Wavelet Level	*Wavelet Sub Band*	*Entropy*
1	HH	3.40
	HL	4.18
	LH	3.69
	LL	8.25
2	HH	4.59
	HL	5.65
	LH	4.95
	LL	9.22
3	HH	6.10
	HL	7.37
	LH	6.32
	LL	9.97
4	HH	7.40
	HL	8.57
	LH	7.62
	LL	9.59
5	HH	7.61
	HL	7.93
	LH	7.55
	LL	7.90

Table 1.1: Entropies of various wavelet sub bands in the Lena decomposition

image, and the per pixel entropy of the sub-bands measured at each level of the decomposition. The values are given in table 1.1. As can be seen from the values, the benefits of applying the wavelet transform with the DAUB16SYM filter past 5 levels for this image are minimal, since the entropies of the wavelet sub-bands are approximately equal.

As previously mentioned, image source coding seeks to find an alternative data representation that is more compact than the original. This is achieved by exploiting the redundancies found in visual data, be it still images or video. In still image compression, spatial redundancy, related to the spatial correlation, is abundant. Indeed, there are large patches of near-constant tone in several places in the *Lena* image. Video has an added dimension - time - and this introduces further redundancies. Since adjacent frames of the video sequence must be similar for the human visual system to perceive motion, there is obvious temporal redundancy between frames. This is often exploited by only sending the frame differences, which are obviously more compact than the full frame representation. A good visual data compression system should exploit redundancy as efficiently as possible whilst also taking into account the characteristics of the human visual system [Wat88], removing only the perceptually least relevant information.

The distortion measure most commonly used in the image coding community is the mean square error, MSE. Given a signal, x_i, and an approximation to the signal, $\hat{x}_i$, the mean square error is defined as

$$d_{MSE}(x, \hat{x}) = \mathcal{E}(x - \hat{x})^2 \tag{1.3}$$

where $\mathcal{E}(\cdot)$ is the expected value. In practice, the sample average is used, i.e.

$$\epsilon = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x}_i)^2 \tag{1.4}$$

The results in this work are presented as peak signal to noise ratios, defined as

$$PSNR = 20 \log_{10} \left(\frac{p}{\sqrt{\mathcal{E}(x - \hat{x})^2}} \right) \tag{1.5}$$

where p is the peak value of the signal. In this work, all images are 8 bits per pixel, giving a maximum value of $p = 255$. MSE is generally regarded as a poor measure of visual quality, since the values are quickly skewed by a small number of outlying values, and the measure does not take into account the properties of the human visual system. However, it is very convenient analytically and can be justified by observing that for small enough errors, $\epsilon << \frac{1}{n}\sum_{i=1}^{n} x_i^2$, any distortion measure will be approximately quadratic.

1.4 The State of the Art

Current visual data compression techniques broadly fall into the following categories :

Predictive Methods Predictive coders are simple to implement and computationally efficient. They have been combined with a multitude of other techniques in order to extend their usefulness [Jai89] [RC95] [LW96]. The main problem with predictive coding is that it is effectively causal, whereas spatial visual data is rarely so. That is, there is no obvious preferred direction of an image, unlike a one dimensional signal. Predictive methods can be generally expressed as

$$\hat{x}(i,j) = \sum_{(k,l)\in\mathcal{N}} a_{kl} q_{i-k,j-l} \tag{1.6}$$

where $\mathcal{N}$ is a, normally causal, spatial neighbourhood of pixel (i,j), a_{kl} is the prediction coefficient of the pixel at position (k,l) in the neighbourhood and q_{ij} are the quantized coefficients to predict from. The prediction coefficients, a_{kl} are either fixed or, in the case of adaptive prediction, estimated locally.

Transform Methods The image data is acted upon by some decorrelating transform. The transform coefficients are then quantized and entropy coded. Since the transform is decorrelating and energy compacting, many of the transform coefficients will be close to zero. Any entropy coding should take advantage of these common occurrences of zeros. Since the transform is non-causal, these methods often achieve greater compression, albeit at the cost of greater complexity. Transform methods attempt to approximate the eigenvector transform of the covariance of a stochastic signal, i.e. the Karhunen-Loève transform. The degree of approximation of the Karhunen-Loève transform will determine the number of zero transform coefficients for a given signal. This, in turn, determines the compression ratio. The currently accepted standard image compression algorithm, JPEG [PM92] [Edi91], is based on the discrete cosine transform [GW92] [Cla95]. The DCT does, however, introduce disturbing blocking artifacts at low bit rates. Transform methods take the form

$$\hat{\boldsymbol{x}} = \boldsymbol{T}^{-1}\boldsymbol{Q}(\boldsymbol{T}\boldsymbol{x}) \tag{1.7}$$

where $\boldsymbol{Q}$ is a quantizer, $\boldsymbol{T}$ is the transform and $\boldsymbol{T}^{-1}$ its inverse. $\boldsymbol{x}$ is a vector representing, for example, an 8×8 block of image data.

Multiresolution Methods Multiresolution methods attempt to identify psychovisually important features at different scales within a signal [BA83]. Once the multiscale decomposition is complete, coding proceeds in a variety of ways, pulling from both predictive and traditional transform methods. The current benchmark for all image coding algorithms is Shapiro's *Embedded Zerotree of Wavelet Coefficients* coder (EZW) [Sha93]. It uses a wavelet transform and encodes a given number of the coefficients using his novel zerotree structure.

Fractal Methods Fractal methods have commonality with all the previous coding methods and also introduce new concepts. Fractal coding methods are traditionally very computationally expensive and will be discussed at length in chapter 2. They were popularised by Jacquin [Jac90] and Barnsley [Bar88] [BD85] [BH92].

Other Techniques Over the years, a variety of ad hoc approaches to compression, often motivated by properties of the visual system, have been put forward (for example Graham's contour coding method [Gra67]). More recently, "model-based" coders using methods derived from computer graphics have been explored [Pea95]. These techniques seldom have the robustness and generality required to deal with the variety of images encountered in all applications.

1.5 Thesis Outline

This thesis is concerned with novel methods of coding still images and video sequences using the wavelet transform and its derivatives.

Chapter 2 provides an overview of the theory of iterated function systems, their application to image compression, popularised by Jacquin [Jac90], and a general review of fractal based image coding schemes. A novel analysis of the underlying fractal basis follows and this leads to the description of the wavelet transform. The chapter concludes with an overview of the state of the art of wavelet based image coding schemes.

In chapter 3, a novel method of exploiting fractal coding methods in the wavelet domain is described. A description of the machinery required for the Karhunen-Loève transform is interspersed with the actual coder description. Following from this, the concept of orientation is introduced and related to the wavelet sub-band decomposition. The effect of introducing the orientation estimate is investigated in the simulation results.

Chapter 4 builds on the orientation concepts in chapter 3 by presenting a novel, complex valued, oriented wavelet transform which retains many of the properties of the traditional wavelet transform which are desirable in an image coding application. A new tree structured vector quantizer algorithm is described, which uses the Linde-Buzo-Gray [LBG80] method of codebook design internally for restructuring. Chapter 4 concludes by combining these new concepts, and the concepts of chapter 3, into a still image coding system. Results are presented for a variety of parameter combinations and compared to current benchmark algorithms.

Chapter 5 presents an overview of video coding techniques, defines the three dimensional, Cartesian separable wavelet transform and describes some of the spatio-temporal properties of the transform. A novel linear quantizer based video coding system is then described and results presented. The shortcomings of this system are outlined and are then, to some degree, addressed by a video coder based on the tree structured vector quantizer of chapter 4. The coder is exercised on a variety of video sequences, at a variety of bit rates. Finally, comparisons are drawn with current video coding techniques.

A review of the novel elements contained in the thesis is presented in the final chapter along with conclusions and possible avenues of further research.

1.6 Reproduction Limitations

This thesis was printed on a Lexmark Optra 2450 laser printer. As a result, the results contained herein are subject to printer artifacts which may be mistaken for coding artifacts.

Chapter 2

Iterated Function Systems and the Wavelet Transform

Image data should naturally contain patches of similar detail at the same and different scales. Consider a forest scene. As the scale at which the scene is viewed changes, certain similarities become apparent. For instance, leaves on the left and right side of a tree would look similar, up to a rotation of π, and from a large distance, a whole tree may appear similar to a leaf viewed close up. The relevance of this form of symmetry in image representation has been discussed at length by Barnsley [Bar88] [BD85] [BH92]. Fractal image compression relies on this premise. To demonstrate the relevance of image symmetry to this method, refer to figure 2.1. The block at the bottom of the image (the *source* block) may be used as a *template* for the other blocks (*target* blocks) shown in the figure. Approximations to the target blocks are created from the source block by appropriate rotations and flips around the mid-point axes. This simple expression of an im-

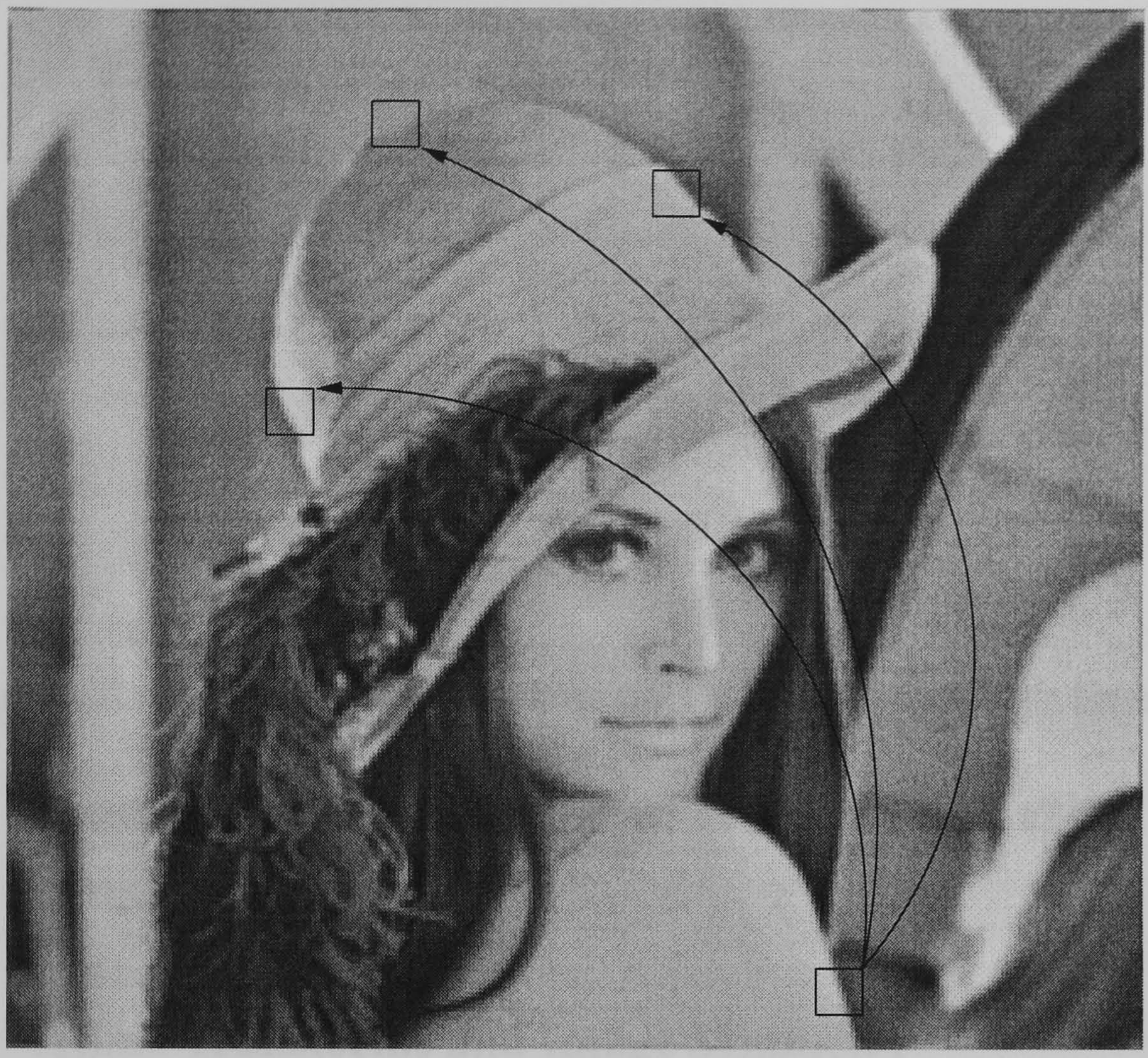


Figure 2.1: Demonstration of image symmetries relevant to fractal block coding

age's self-similarity is the basis for all fractal image compression methods [Jac90] [Bar88] [BD85] [BH92].

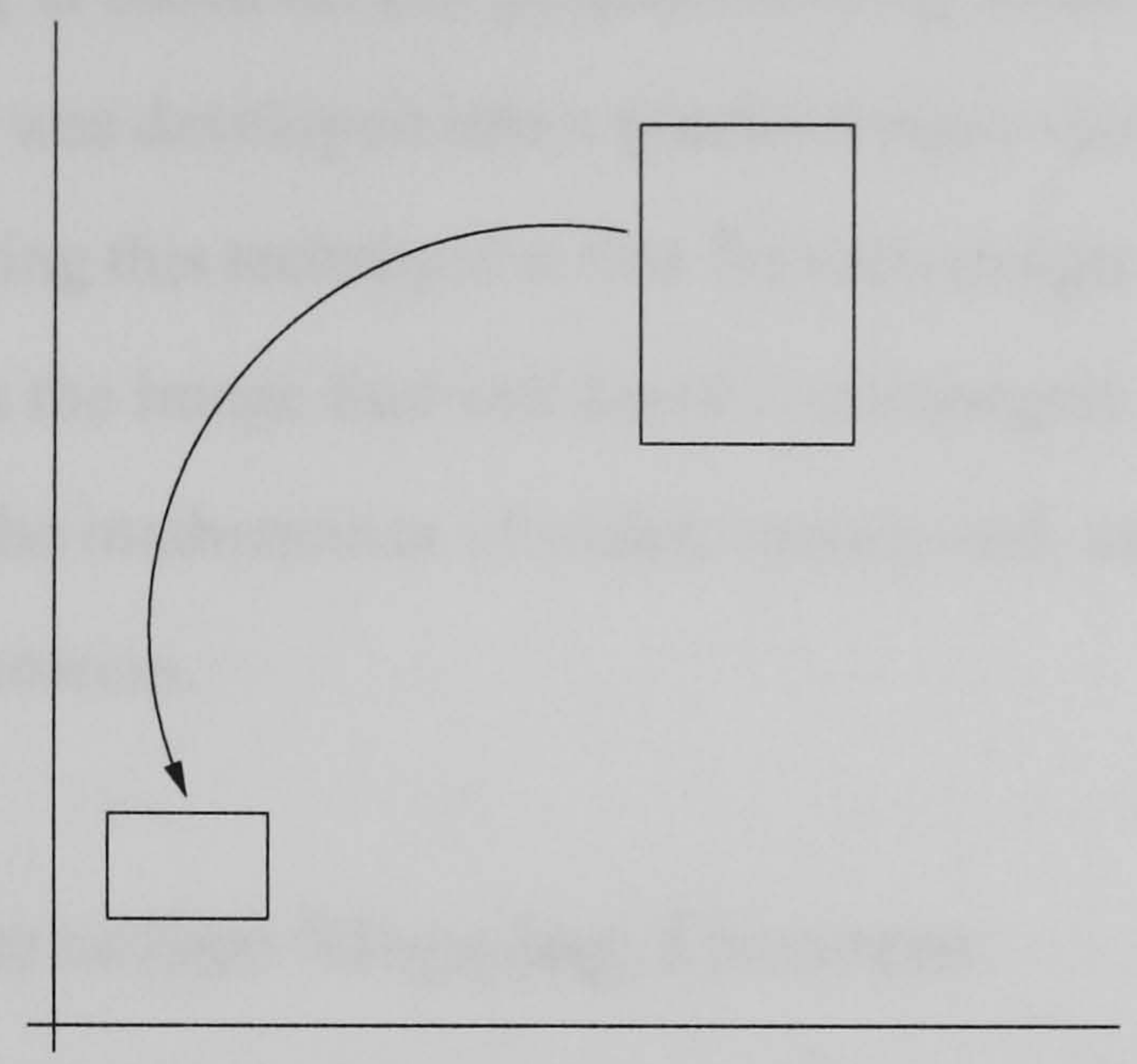


Figure 2.2: A representative transform from the required affine subgroup

2.1 Fractal Block Coding and Iterated Function Systems

2.1.1 Affine Transforms and Image Symmetry

An affine transform in n dimensions is one of the form

$$\boldsymbol{T}:\boldsymbol{X}\rightarrow\boldsymbol{X},\boldsymbol{T}\boldsymbol{x}=\boldsymbol{L}\boldsymbol{x}+\boldsymbol{t}=\begin{bmatrix} l_{11}\dots l_{1n} \\ \vdots\vdots \\ l_{n1}\dots l_{nn}\end{bmatrix}\begin{bmatrix} x_1 \\ \vdots \\ x_n\end{bmatrix}+\begin{bmatrix} t_1 \\ \vdots \\ t_n\end{bmatrix} \quad (2.1)$$

For the purposes of fractal image compression, a subgroup of the affine group, consisting only of rotations, scalings and translations is required. Figure 2.2 shows a representative transform. Note that the transform reduces the size of the block - it is contractive. This is a further limitation on the choice of the blockwise transforms.

Fractal block coding is based on the ground-breaking work of Barnsley [Bar88] [BD85] [BH92] and was developed into a practical algorithm by Jacquin [Jac90]. The concept underlying this technique is that for each image, there exists a block-wise transform upon the image that will leave it unchanged. The roots of fractal block coding lie in the mathematics of metric spaces and, in particular, the Contraction Mapping Theorem.

2.1.2 The Contraction Mapping Theorem

Definition 2.1.1 (Contraction Mapping) *Let (X, d) be a complete metric space. That is let X be a vector space over some field $\mathcal{F}$ and d be a metric upon X. Then, a transformation T, $T : X \to X$ is said to be a* contraction mapping *if and only if*

$$\exists s \in \Re, 0 \leq s < 1 \;\; s.t \;\; d(T(x), T(y)) \leq s \cdot d(x, y) \; \forall x, y \in X. \tag{2.2}$$

Here, s is known as the *contractivity factor* of the transformation T.

Theorem 2.1.1 (Contraction Mapping Theorem) *For a contraction mapping T on a complete metric space X, there exists a* unique *fixed point $x^\star \in X$ such that $x^\star = T(x^\star)$. The unique fixed point is then*

$$x^\star = \lim_{i \to \infty} T^i(x_0) \; , x_0 \in X$$

For proof and the Collage theorem, see Appendix A.

Theorem 2.1.1 states that if a contraction mapping is iterated, there exists a point where further application of the mapping has no effect. The point at which this occurs is called the fixed point of the contraction mapping and, remarkably, is independent of the initial conditions of x_0. Define an *Iterated Function System (IFS)* [BD85] [Bar88] [BH92] as a set of contraction mappings $W = \{w_n : n = 0, \ldots, N\}$, $w_n : X \to X$ with contractivity factors s_n. Then, the IFS is itself a contraction mapping on X and has contractivity factor $s = max\,\{s_n\}$. Note that, implicit to the IFS definition, is the domain and range of each map. Each map has distinct domain on X, usually a compact, non-empty subset of X. Now, since the IFS is a contraction mapping on X, it too has a unique fixed point $x^\star \in X$. Since $x^\star$ is unique, it is completely specified by the IFS, W. The image processing problem, termed the *inverse problem* by Barnsley [BH92] is then *"Given an image I, can an IFS, W, be found such that the fixed point of W is I?"* At the present time, there is no general method for finding such an IFS, given I. However, using the IFS fixed point equation

$$x^\star = W(x^\star) = w_1(x^\star) \cup \cdots \cup w_n(x^\star), \tag{2.3}$$

where $\cdot \cup \cdot$ represents the 'pasting together' of transformed patches, and the Collage Theorem (section A.2)

$$d(I, x^\star) \leq \frac{1}{1-s} d(I, W(I)) \tag{2.4}$$

an approximation method can be constructed. Equation 2.3 states that to find the IFS W, a set of contractive transforms should be applied to I and the resultant patches *pasted* together to reconstruct I. The uniqueness of $x^\star$ is important since

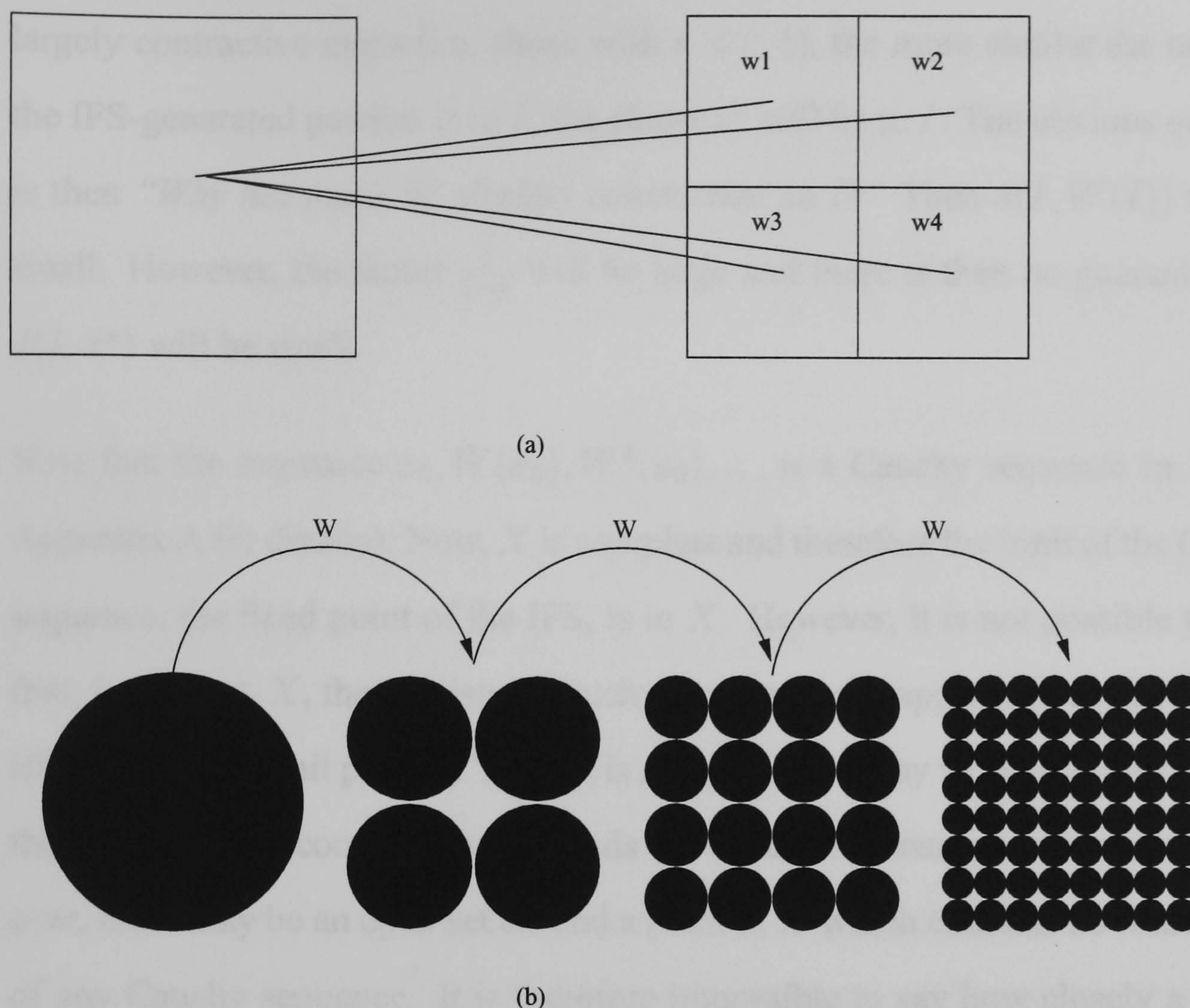


Figure 2.3: Construction of an IFS and the initial stages of its iteration

if $I = W(I)$, it is known that $I = x^{\star}$, that is I is the fixed point of the IFS W. Figure 2.3 shows the process of constructing an IFS and a resultant application. In figure 2.3(a), the IFS consisting of the maps $\{w_1, \ldots, w_4\}$ exactly covers the square. (Each map is simply an averaging on the square in each axis.) From the previous discussion, the IFS $\{w_1, \ldots, w_4\}$ will *exactly* generate the square from any initial condition. Figure 2.3(b) shows the first three iterations of the IFS on a circle. It is apparent that, in the limit, the IFS will indeed converge to a square. Equation 2.4 states that if an exact mapping cannot be found to create I, then, for largely contractive maps (i.e. those with $s << 1$), the more similar the union of the IFS-generated patches is to I, the closer $x^\star$ will be to I. The obvious question is then *"Why not make W slightly contractive on I?"* Then $d(I, W(I))$ will be small. However, the factor $\frac{1}{1-s}$ will be large and there is then no guarantee that $d(I, x^\star)$ will be small.

Note that the sequence $x_0, W(x_0), W^2(x_0), \dots$ is a Cauchy sequence in X(see Appendix A for details). Now, X is complete and therefore the limit of the Cauchy sequence, the fixed point of the IFS, is in X. However, it is not possible to state that, for all $x \in X$, there exists a Cauchy sequence of mappings (and therefore an IFS) in X with limit point x. Thus, it is not possible to say with absolute certainty that fractal image compression methods can exactly reconstruct any image. Moreover, there may be an open set around a point in X which contains no limit points of any Cauchy sequence. It is therefore impossible to say how closely a fractal image compression system can approximate an arbitrary image. This represents a fundamental weakness in a general purpose image coding method.

2.1.3 Fractal Block Coding

Fractal block coding, as described by Jacquin [Jac90], is a notoriously slow and computationally intensive process. Given an arbitrary image $\mathcal{I}$, it is partitioned into non-overlapping blocks, $\{D_j\}$ of size $d \times d$. These blocks will be known as *domain blocks*, following the notation[1] of Barnsley [BH92]. The image is then

[1] Barnsley's notation is slightly confusing since it refers to the *inverse* transform.

Ordinal	*Isometry*
1	Identity
2	Flip along mid-X axis
3	Flip along mid-Y axis
4	Flip along major diagonal

Table 2.1: Base isometry set used in conventional fractal block coders

spatially averaged by 2 in both the horizontal and vertical directions, extracting all, possibly overlapping, blocks of size $d \times d$ producing the pool of *range blocks*, $\{R_i\}$. The goal of a fractal block coder is then to find the optimal parameter set $\{a, b, c, i\}$ for each block D_j in the approximation

$$\hat{D}_j = a \cdot \iota_c(R_i) + b = T_j(R_j) \tag{2.5}$$

such that the error $d(D_j, \hat{D}_j)$ is minimised. It is usual for d to be the metric derived from the L_2 norm, $||D_j - \hat{D}_j||^2$, the squared error measure. The set $\{\iota_i\}$ is the set of isometries generated by the dihedral group of the square. Table 2.1 shows the base isometries which are combined to form the eight isometries of the square. The fractal block code for the image then consists of the parameter set $\{a, b, c, i\}$ for each domain block D_j in the domain pool $\{D_j\}$. The parameter sets are entropy coded to provide compression. Note that the parameter set $\{a, b, c, i\}$ is restricted since the final transform $T = \bigcup_j T_j$ must be a contraction mapping under the metric d. Jacquin [Jac90] notes that the isometries all have unity L_2-contractivity and the contrast scale by a has L_2-contractivity of a^2. Hence, to ensure that the transform for each block is a contraction mapping, the value of a must be restricted such that $0 \leq a < 1$. This, in turn, limits the contrast scalings to dynamic range reductions. Then, if all the block transforms are contraction mappings, the union of the block transforms is itself a contraction mapping.

Reconstruction of the approximation to the image $\mathcal{I}$ from the fractal block code follows from the Contraction Mapping Theorem and the Collage Theorem. Note that $T = \bigcup_j T_j$ is, by the Collage Theorem, a contraction mapping. Therefore, by the Contraction Mapping Theorem, T has a unique fixed point, $x^\star = \lim_{n\to\infty} T^n(x_0)$. Since the blockwise component contraction mappings were chosen to minimize the individual errors, their union minimizes the overall error. Hence, the unique fixed point will, in some sense, be *close* to $\mathcal{I}$. Reconstruction, therefore, is simply a matter of iterating the maps on any initial image, spatially averaging by two between the iterations. The choice of initial image is irrelevant since, by the Contraction Mapping Theorem, the fixed point x^* is unique. Therefore, $\forall x \in X, \lim_{x\to\infty} T^n(x) = x^*$. The iterative reconstruction process is continued until some specified error condition has been met or until the difference between successive iterations is below some threshold.

2.1.4 Properties and pitfalls of the fractal method

Fractal image compression has suffered, from the outset, claims of extreme compression ratios [BH92]. These extreme claims have come about because of the resolution independence of the fractal block coding method. That is, it is possible to generate the fractal block code for an image of 256×256 pixels and reconstruct it at 1024×1024 pixels (since the maps are resolution independent), resulting in an apparent increase of 16 times in the compression ratio. The resolution independence of the method also allows so called *infinite zooming*. Again, a zoomed area is reconstructed at a higher resolution. The iterative method will add false detail to the zoomed image, making it more pleasing to the casual observer. However, it is important to note that any detail added is, of course, interpolated false detail : it increases the overall error.

Moreover, fractal image compression is a highly asymmetric process; that is, coding takes much longer than decoding. This is due to the extensive nature of the domain block search. This can be a disadvantage in some applications.

2.2 Advances in Fractal Block Coding Algorithms

Much research has revolved around speeding up the coding in some way. Jacquin [Jac90] suggested classifying the range blocks into three distinct classes: shade blocks, midrange blocks and edge blocks. The coder then only checks range blocks in the same class as the current domain block when searching for the optimal transform. Details of the classification algorithm used by Jacquin can be found in [RA86].

Jacobs, Fisher and Boss [JFB92] classify blocks into 72 different classes and also apply a quadtree partitioning scheme. The quadtree scheme works by using larger blocks (32×32 in their paper) and splitting the block into four smaller blocks should an error condition not be satisfied. This quadtree decomposition is repeated until the error condition is satisfied or a minimum block size is reached. This has obvious advantages if parts of the image contain large regions of relatively constant greyscale.

The same authors [JFB91] proved that not all the w_i in the IFS $\{X; w_0, \ldots, w_n\}$ need to be contractive. They define a map $W : F \to F$ as *eventually contractive* if, for some $n \in \mathcal{Z}^+$,W^n is contractive. They then prove that the fractal decoder will converge if $T = \cup_i T_i$ is only eventually contractive. Here, the iterated transform T^m is the composition of transform unions of the form

$$w_{i_1} \circ w_{i_2} \circ \ldots \circ w_{i_m}.$$

Since the contractivities of each union w_{i_j} multiply to give the overall contractivity of the iterated transform, the composition may be contractive if it contains sufficiently contractive w_{i_j}. Intuitively, it is simple to see that if the union consists of slightly expansive transforms and highly contractive ones, then the union will eventually be contractive. The provision for eventually contractive maps allows the coder to achieve better results. Since there is now no longer a bound on the contractivity of the component transform , the dynamic range of range blocks may now be *increased* to be similar to that of the domain block.

Most speed-ups have reduced the size of the pool that the coder must search for each domain block in some way. Reducing the size of the search pool, however, can have an adverse effect on reconstructed image quality since the range block pool is not as rich. Saupe [Sau95] uses the theory of multi-dimensional keys to perform an approximate nearest-neighbour search. Since this search can be completed in $O(logN)$ time, the range pool need not be depleted to achieve a speed up. Saupe's basic idea is that of a $(d-1)-$dimensional projection on $\Re^d$ where $d \geq 2$. He defines a subspace, $X \subseteq \Re^d$ as $X = \Re^d \backslash \{r \cdot e : r \in \Re\}$ where $e = \frac{1}{\sqrt{d}}(1, \ldots, 1) \in \Re^d$. Defining $< \cdot, \cdot >$ to be the inner product operator and a normalised projection operator $\phi : X \to X$ and a function $D : X \times X \to [0, \sqrt{2}]$ by

$$\phi(x) = \frac{x - < x, e > e}{||x - < x, e > e||}$$

and

$$D(x, z) = \min(d(\phi(x), \phi(z)), d(-\phi(x), \phi(z)))$$

gives an expression for the squared error

$$E(x, z) = < z, \phi(z) >^2 g(D(x, z))$$

where $g(D) = D^2(1 - D^2/4)$. This theorem states that the squared error, $E(x, z)$ is proportional to $g(D)$ which is a simple function of the Euclidean distance between $\phi(x)$ and $\phi(z)$ (or $-\phi(x)$ and $\phi(z)$ since $\phi(x)$ is unique up to sign). Note also that g is monotonically increasing on the interval $[0, \sqrt{2}]$. Saupe then states that the minimisation of the squared errors $E(x_i, z)$ $i = 1, \ldots, N$ is equivalent to the minimisation of the $D(x_i, z)$. Thus, the squared error minimisation may be replaced by the nearest neighbour search for $\phi(z) \in \Re^d$ in the set of $2N$ vectors $\pm\phi(x_i) \in \Re^d$. Since $\Re^d$ is a Euclidean space, any of the nearest neighbour search algorithms that run in expected logarithmic time, for example kd-trees [FJF77], can be applied. Saupe does, however, note that an 8×8 block results in 64 dimensions for the multi-dimensional keys. He therefore suggests downsampling the blocks to, say, 4×4 which reduces storage requirements to just 16 dimensions.

Other speed up methods have revolved around performing little or no searching for the range blocks. Monro et al [MWN93] [MDW90] [Mon93] [MD92] , have developed and patented a technique known as the *Bath Fractal Transform* [2] (BFT). The BFT works by limiting the search of the range blocks and also using higher order functions on the blocks. Mathematically, if $W = \{X; w_k, k = 1, \ldots, N\}$ is an IFS with attractor A, they define a *fractal function* f on A as $f(w_k(x,y)) = \nu_k(x, y, f(x,y))$, where the maps ν_k have parameters $\alpha_i^{(k)}, k = 1, \ldots, N, i = 1, \ldots M$. Then, M is the *order* of the IFS and N is the number of free parameters of the BFT. The function f is found by minimising $d(g, \hat{g})$ for some suitable metric d over block k where

$$\hat{g}(x) = \nu_k(w_k^{-1}(x), g(w_k^{-1}(x))),$$

g is the original image greyscale values and $\hat{g}$ is the approximation. Solving

$$\frac{\partial d(g, \hat{g})}{\partial \alpha_i^{(k)}} = 0 \;\; \forall i, k$$

gives the BFT. They define various searching options for the BFT, defined before downsampling has occurred; that is, they assume that domain blocks are $d \times d$ while range blocks are $2d \times 2d$. A level zero search chooses the range block in the same position as the domain block. A level one search would include the four range blocks that intersect the domain block. A level two search would also include all range blocks that intersect those in level one and so on. The complexity options used, however, make the BFT unique. As it stands, such a limited search would severely degrade reconstructed image quality. Allowing higher order terms

[2]They are at Bath University, England.

in the BFT reduces the error while keeping encoding times low. At its most basic level (order zero), the BFT degrades to Jacquin's method[Jac90]. That is, the maps ν_k have the form

$$\nu_k(x, y, f) = s_k \cdot f + t_k.$$

An order three complexity gives the maps the form

$$\nu_k(x, y, f) = a_3^{(k)} x^3 + a_2^{(k)} x^2 + a_1^{(k)} x + b_3^{(k)} y^3 + b_2^{(k)} y^2 + b_1^{(k)} y + s_k f + t_k.$$

Note that there are no cross product terms (for example xy or $x^2 y$) so that calculation is kept relatively simple. They recently [MW94] developed a proprietary reconstruction algorithm, the *Accurate Fractal Rendering Algorithm* (AFRA) ,which remains unpublished at this time, specifically designed for use with the BFT.

Barthel and Voyé [BV94] modified the luminance transform to act in the frequency domain. They propose the following high-order luminance transform.

$$\lambda(g) = IDCT\left(\bigcup_{u=0}^{N-1}\bigcup_{v=0}^{N-1} a(u,v) \cdot G(u,v) + b(u,v)\right)\ , \ G(u,v) = DCT(g)$$

where DCT denotes the Discrete Cosine Transform and $IDCT$ denotes the inverse discrete cosine transform, N is the size of the blocks and g is the range block itself. If the DCT of the domain block f is denoted by $F(u,v)$, the spectrum of the domain block $F(u,v)$ can be approximated by scaling of the spectrum of $G(u,v)$. They conjecture that most blocks can be sufficiently well approximated by a low order transform, negating the risk of a bit-explosion due to the excessive number of parameters to be coded. Further to this, they propose modifying the luminance transform so that it operates on a frequency domain partition. The luminance transform would then be expressed as

$$\lambda_k(g) = IDCT\left(\bigcup_{u=0}^{N-1}\bigcup_{v=0}^{N-1}\begin{cases} a_0 \cdot G(u,v) + b & \text{if } u = 0, v = 0 \\ a(u,v) \cdot G(u,v) & \text{otherwise} \end{cases}\right)$$

where $a(u, v) = a_i$ if $(u, v) \in R_i$ $i = 1, \ldots, K$. This permits their modified frequency domain luminance transformation to be used in a block splitting procedure. This is similar to quadtree partitioning, but if the top level block does not satisfy the error condition, they only recode the sub-blocks which do not satisfy the error condition. In this way, they can achieve similar reconstruction results to quadtree partitioning but with fewer transform coefficients.

Gharavi-Alkhansari and Huang [GAH94] use a combination of fractal coding and basis block projection. For each domain block, they generate three pools of range blocks thus

i *Higher Scale Adaptive Basis Blocks*

This pool is the standard range block pool from a normal fractal coder. That is, spatially averaged copies of the domain blocks, augmented by rotated and reflected versions.

ii *Same Scale Adaptive Basis Blocks*

This pool is generated by selecting regions of the image which are the same size as the domain blocks. These blocks, however, must be selected causally, that is they may only be selected from parts on the image which have already been encoded. This pool may also be augmented by rotations and reflections.

iii *Fixed Basis Blocks*

This is a fixed pool of basis blocks that are known *a priori* to both the encoder and decoder. They need not be orthogonal or even complete.

The purpose of the fixed basis blocks is to allow the encoder to accurately encode a domain block that is dissimilar to any range block in the image. However, the lack of an orthogonality condition on the basis blocks appears to make the optimal solution for the coefficients of the basis blocks a difficult task. The authors offer two sub-optimal methods of calculating basis block coefficients. The first is to select the basis block most strongly correlated with the domain block, then orthogonalise the domain block with respect to the basis block and repeat until an error condition is satisfied or all basis blocks have been used. The second, more general method, is to do the same as the first method, but also orthogonalise all other basis blocks with respect to the most correlated. They also note that standard fractal block coding, block transform coding and vector quantisation are all special cases of their proposed algorithm. As might be expected, their algorithm is expensive computationally.

2.3 Connection with Conventional Coding Methods

Fractal image compression, in all its forms, can be seen as a mapping from image to image. Conventional predictive image coding [Jai89] is also defined as a mapping from image to image of the form

$$f = Af + b \tag{2.6}$$

where A is a linear predictor matrix and b is the prediction error. In conventional linear prediction, A is a causal predictor. That is, A relies only on data previously coded to form its predictions. The prediction function is often modified on a per-pixel basis in order to obtain the best prediction at any point. Since it is causal, the same prediction operation can be performed at both the encoder and decoder and so the predictions generate no extra information. Most computation in conventional predictive coding is concentrated on encoding b, the prediction error, as compactly as possible. Prediction in this framework is usually on a per-pixel basis. Now, fractal image compression works in much the same way, although the prediction is applied to blocks and the predictor is modified on a per-block basis and consequently some data must be sent to encode the predictor parameters. The parameters in fractal coding determine the position and scale factor of the prediction. Re-writing equation 2.6 as

$$\hat{D}_j = A_j R_i + \sum_k d_{jk} B_k \tag{2.7}$$

where B_k is a basis of the underlying space and the prediction is given by $A_j R_i$, fractal image compression can be restated as a subset of conventional predictive methods.

Image data compression methods can be analysed from the viewpoint of the basis that they use to represent the data. The conventional linear predictive method, detailed above, trivially has a complete basis generated by the impulse values from the error term b in equation 2.6 or the error correction terms $d_{jk}B_k$ in equation 2.7. What is the basis presented by conventional blockwise fractal compression methods? Fractal compression proceeds via averaging, dyadic downsampling and shifting to form the prediction pool and addition of constant blocks to handle errors. The characteristics of the basis depend solely on the block size. If the block size is one pixel then, trivially, the system can encode any signal as a linear combination of impulses. In this case, the basis is (over) complete. If the block is the same size as the image , then, again trivially, the basis is constant. Fractal block coding lies somewhere between these two extremes. Figure 2.4 shows the construction of the basis inherent in 1-D fractal coding. The basis is generated via the coding process, by the scaling of dyadically down-sampled blocks, followed by addition of a constant block. Figure 2.4(a) shows a piecewise constant signal on the interval $[a, b]$. Figure 2.4(b) shows the signal dyadically down-sampled and figure 2.4(c) shows these down-sampled blocks with a constant block added. Starting from a piecewise constant signal, the basis is generated by repeated down-sampling and shifting. However, for a given IFS approximation of a signal, only certain shifts are actually used. In general, therefore, the basis cannot be complete. As the process is iterated, the blocks will be repeatedly down-sampled, scaled and

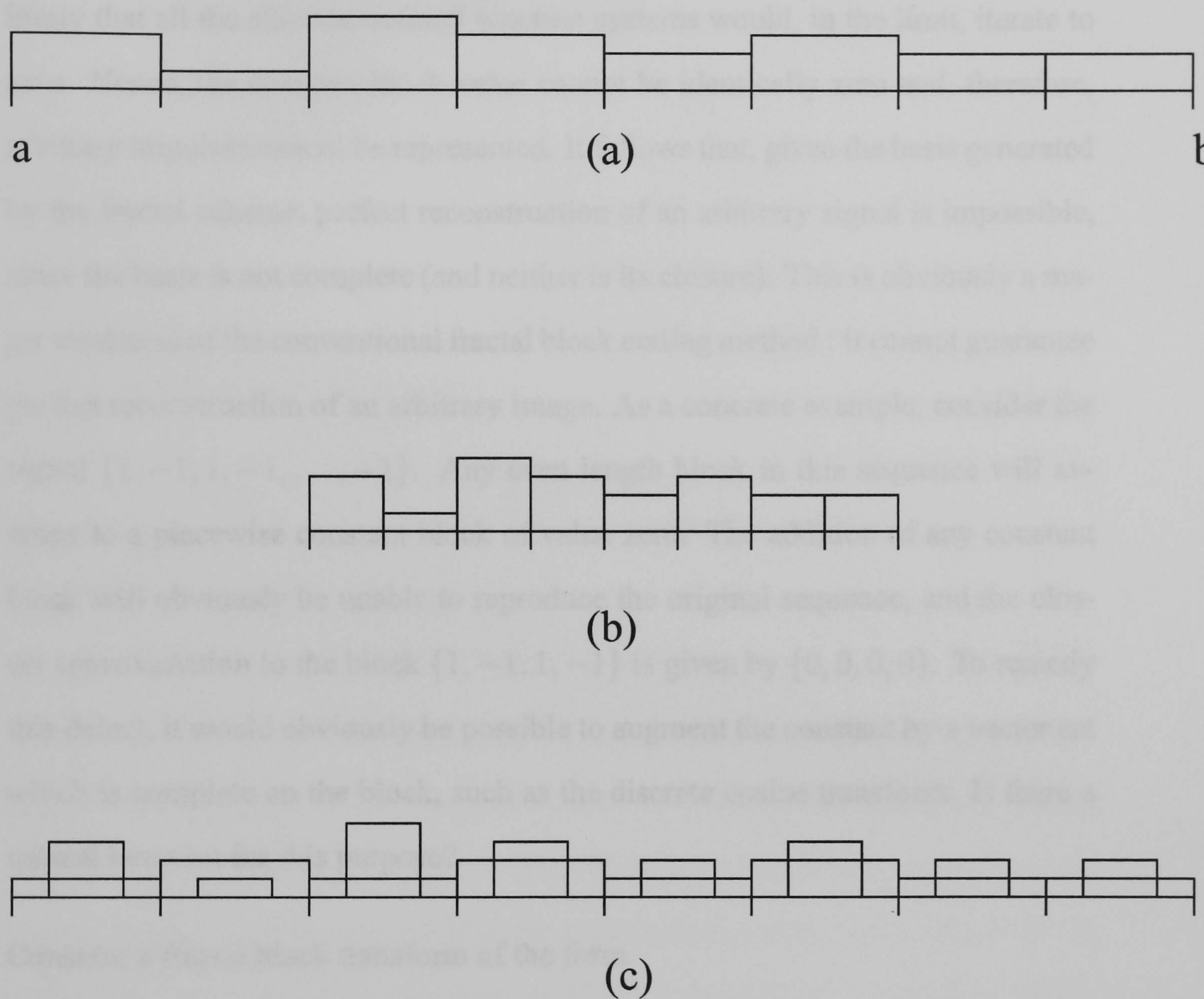


Figure 2.4: The basis created by the fractal block coding scheme

a constant block added. Now, since the scale factor used must be strictly less than unity, the constant value added cannot be identically zero. If it were, the energy in the blocks would decay to zero as the iteration progressed. This, in turn, would imply that all the allowed iterated function systems would, in the limit, iterate to zero. Hence, the constant block value cannot be identically zero and, therefore, arbitrary impulses cannot be represented. It follows that, given the basis generated by the fractal scheme, perfect reconstruction of an arbitrary signal is impossible, since the basis is not complete (and neither is its closure). This is obviously a major weakness of the conventional fractal block coding method : it cannot guarantee perfect reconstruction of an arbitrary image. As a concrete example, consider the signal $\{1, -1, 1, -1, \ldots, -1\}$. Any even length block in this sequence will average to a piecewise constant block of value zero. The addition of any constant block will obviously be unable to reproduce the original sequence, and the closest approximation to the block $\{1, -1, 1, -1\}$ is given by $\{0, 0, 0, 0\}$. To remedy this defect, it would obviously be possible to augment the constant by a vector set which is complete on the block, such as the discrete cosine transform. Is there a natural basis set for this purpose?

Consider a fractal block transform of the form

$$f(x) = \alpha f(\frac{x}{2} - k) + e(x). \tag{2.8}$$

Assuming that $f \in L^2$, this can be expressed as

$$\sum f_{i,j}\psi_{i,j}(x) = \alpha \sum f_{i,j}\psi_{i,j}(\frac{x}{2} - k) + e(x) \tag{2.9}$$

where $\psi_{i,j}$ is some suitable orthogonal basis of L^2. Now, define $\psi_{i,j}$ to be

$$\psi_{i,j} = \psi(\frac{x}{2^i} - j2^i) \tag{2.10}$$

which, if $k' = k2^{2i}$, implies

$$\begin{aligned} \psi_{i,j}(\frac{x}{2} - k') &= \psi\left(\frac{\frac{x}{2}-k'}{2^i} - j2^i\right) \\ &= \psi\left(\frac{x}{2^{i+1}} - \frac{k'}{2^i} - j2^i\right) \\ &= \psi\left(\frac{x}{2^{i+1}} - \frac{k2^{2i}}{2^i} - j2^i\right) \\ &= \psi\left(\frac{x}{2^{i+1}} - 2^i(j+k)\right) \\ &= \psi_{(i+1),(j+k)}(x). \end{aligned} \tag{2.11}$$

Substituting (2.11) into (2.9) gives

$$\begin{aligned} \sum f_{i,j}\psi_{i,j}(x) &= \alpha \sum f_{i,j}\psi_{(i+1),(j+k)}(x) + e(x) \\ &= \alpha \sum f_{(i-1),(j-k)}\psi_{i,j}(x) + e(x) \end{aligned} \tag{2.12}$$

Therefore,

$$f_{i,j} = \alpha f_{(i-1),(j-k)} + e_{i,j} \tag{2.13}$$

This states that, if the basis generated by equation 2.10 is used, the coefficients of f_i should be predicted from the coefficients of f_{i-1} by the fractal block coding algorithm. Furthermore, if the basis $\{\psi_{i,j}\}$ is an orthonormal one, then the scale spaces f_i are all orthonormal. Thus, the error coefficients $e_{i,j}$ are orthogonal for each i. This treatment does, however, require that the shift, k' is a multiple of 2^{2i}. This restriction does not practically challenge the validity of the argument due to the shift invariance and stationarity of the data. The next section makes rigorous the construction and properties of the basis $\psi_{i,j}$.

2.4 The Wavelet Transform

The Fourier Transform method is a well known tool for signal analysis. The Fourier Transform expresses any square integrable 2π periodic function on $L^2([0, 2\pi])$ as the projection of it onto the orthonormal basis

$$\omega_n(x) = e^{\imath nx}, \quad n = \ldots, -1, 0, 1, \ldots.$$

However, if $\omega(x) = e^{\imath x}$ then $\omega_n = \omega(nx)$. Hence, the orthonormal basis $\{\omega_n\}$ is actually the set of integer dilates of the single function ω. Note that

$$\omega(x) = e^{\imath x} = \cos x + \imath \sin x.$$

A remarkable fact about the Fourier Transform is that this is the *only* function required to generate *all* 2π periodic square integrable functions. One problem with the Fourier transform is that of locality. Fourier basis functions are localised only in frequency, not in time. This implies that any errors in the Fourier transform coefficients will produce errors that are localised in frequency, but which spread throughout the whole time domain. This property is one which has adverse effects on image coding schemes. A transform which has basis functions localised in both frequency and time would be preferable [Wil87].

2.4.1 The Continuous Wavelet Transform

The space $L^2(\Re)$ is practically more useful than $L^2([0, 2\pi])$ [Mal89]. Functions in $L^2(\Re)$ satisfy the following condition

$$\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty.$$

Since every function in $L^2(\Re)$ must decay to zero at $\pm\infty$, the Fourier basis functions ω_n, which are sinusoids, are not in $L^2(\Re)$. A synonym for sinusoids is *waves* and if it is required that waves generate $L^2(\Re)$, they must decay very quickly to zero (for all practical purposes). Hence, small waves or *wavelets* are needed to generate $L^2(\Re)$. As in the Fourier Transform case, the wavelets should ideally be generated by one function. However, if the wavelets must decay to zero, the function dilates must be translated along $\Re$ by integral shifts in order to cover $\Re$. Given a *mother wavelet* $\psi \in L^2(\Re)$ of unit energy, then all the translated dilates

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k) \ , \ j, k \in \mathcal{Z}$$

also have unit energy.

Definition 2.4.1 (Wavelet) *A function $\psi \in L^2(\Re)$ is called a* wavelet *if the family $\{\psi_{j,k}\}$ defined by*

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k) \ , \ j, k \in \mathcal{Z}$$

is an orthonormal basis of $L^2(\Re)$. That is

$$< \psi_{j,k}, \psi_{l,m} >= \delta_{j,l}\delta_{k,m} \ , \ j, k, l, m \in \mathcal{Z}$$

where

$$\delta_{j,k} = \begin{cases} 1 & \textit{if } j = k \\ 0 & \textit{otherwise} \end{cases}$$

Then, every $f \in L^2(\Re)$ *can be written as*

$$f(x) = \sum_{j,k=-\infty}^{\infty} c_{j,k}\psi_{j,k}(x). \tag{2.14}$$

The series representation of f in 2.14 is called the *wavelet series* of f and is analogous to the Fourier series of a function. Similarly, the *integral wavelet transform* [Chu92] may be defined, analogous to the continuous Fourier Transform as

$$(W_\psi f)(b,a) = |a|^{-1/2} \int_{-\infty}^{\infty} f(x)\psi\left(\frac{x-b}{a}\right) dx, \quad f \in L^2(\Re).$$

Wavelets are localised both in frequency (scale) by dilations *and* in time by translations. Errors in the coefficients will produce errors which are localised in both frequency and time, unlike the Fourier transform as detailed in section 2.4.

2.4.2 The Discrete Wavelet Transform

The continuous wavelet transform is a useful theoretical tool, but image processing deals with sampled images, i.e discrete values. A simple and fast method of calculating the wavelet coefficients of a signal is required. Daubechies [Dau88] defines the 1-D discrete wavelet transform in terms of a pair of *quadrature mirror filters* (QMF pairs) [Dau92] [AH92]. If $\{g_n\}$ is a sequence that is generated by sampling the mother wavelet ψ, then g will be a high pass filter. The mirror filter of this is $\{h_n\}$, given by the relation $g_m = (-1)^m h_{1-m}$. The discrete wavelet transform is then defined in terms of successive filtering and dyadic downsampling.

$$\begin{aligned} c_n^i &= \textstyle\sum_k h_k c^{i-1}(2n-k) \\ d_n^i &= \textstyle\sum_k g_k c^{i-1}(2n-k) \end{aligned}$$

where $c^0(n) = f(n)$ is the original signal and the wavelet transform coefficients consist of $\{d_n^i\}$. For reconstruction, it is usual to zero pad the data with $2^i - 1$ zeros at level i. Defining a family of filters $\{h_n^i\}$ given by

$$h_k^i = \begin{cases} h_n & \text{if } k = 2^i n \\ 0 & \text{otherwise} \end{cases}$$

and letting $f^i(n)$ and $e^i(n)$ be the zero padded data at level i, then the reconstruction from level i to level $i-1$ is given by

$$f^{i-1}(n) = \sum_k h_{-k}^{i-1} f^i(n-k) + \sum_k g_{-k}^{i-1} e^i(n-k).$$

The path from the 1-dimensional transform to the 2 dimensional version is simple. With the correct filters, the above definition of the discrete wavelet transform becomes separable in 2 dimensions. That is, the 1 dimensional wavelet transform may be performed on the rows, followed by the columns.

Figure 2.5 shows the process of a 2 dimensional wavelet decomposition. Figure 2.6 is the corresponding reconstruction process.

In her landmark paper, Daubechies [Dau88] developed a set of filters for orthonormal, compactly supported wavelets of differing sizes. These particular wavelet

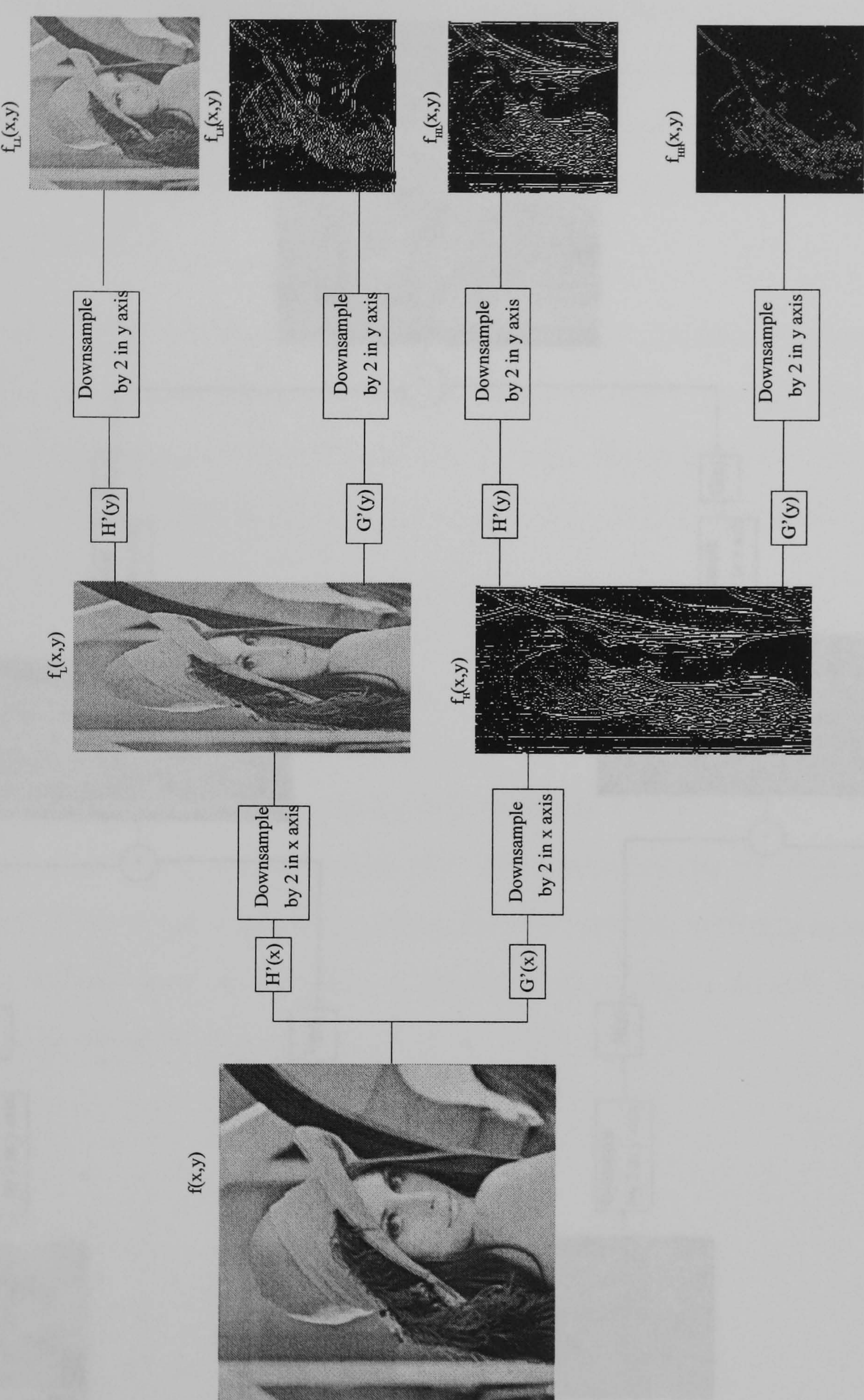


Figure 2.5: Block diagram of a 2 dimensional wavelet decomposition

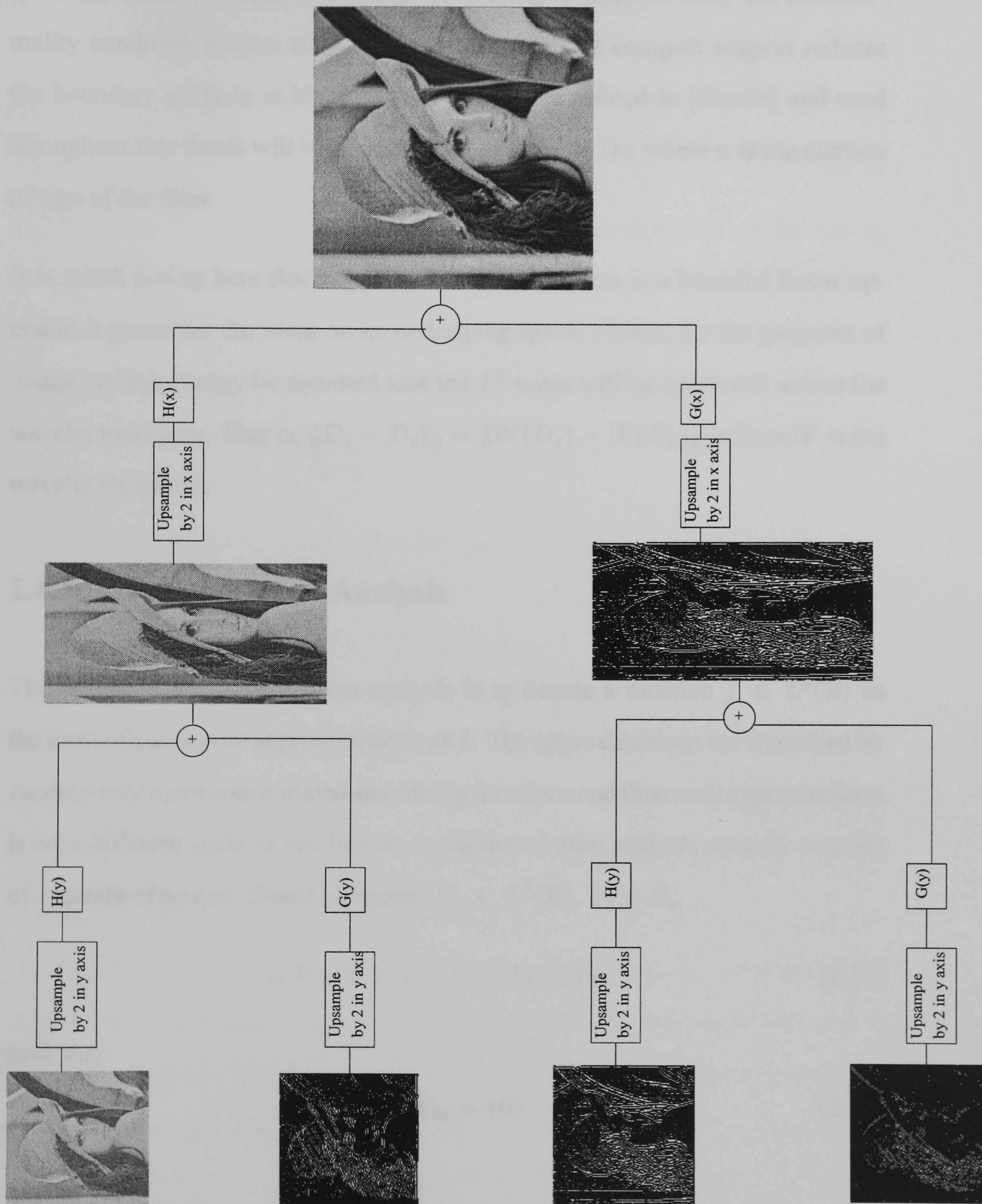


Figure 2.6: Block diagram of a 2 dimensional wavelet reconstruction

bases are ideal for block-wise image processing techniques since the orthonormality condition assures easy computation, while the compact support reduces the boundary artifacts at block edges. The filters defined in [Dau88] and used throughout this thesis will be denoted by $DAUBn$ or Dn where n is the number of taps of the filter.

It is worth noting here that since the wavelet transform is a bounded linear operator, it preserves the norm on its underlying space. Hence, for the purposes of image coding, it may be assumed that the L^2 norm will be preserved across the wavelet transform. That is, $||D_1 - D_2||_2 = ||W(D_1) - W(D_2)||_2$ where W is the wavelet transform.

2.4.3 Multiresolution Analysis

The notion of a multiresolution analysis is to denote a function $f \in L^2(\Re)$ as the limit of successive approximations of f. The approximations are smoothed by successively more concentrated smoothing functions and thus each approximation is on a different scale or resolution. A multiresolution analysis actually consists of a family of nested, closed subspaces $V_m \subset L^2(\Re)$, $m \in \mathcal{Z}$,

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \tag{2.15}$$

such that

$$\bigcap_{m \in \mathcal{Z}} V_m = \{0\} \tag{2.16}$$

and

$$\overline{\bigcup_{m \in \mathcal{Z}} V_m} = L^2(\Re) \tag{2.17}$$

and

$$f \in V_m \Leftrightarrow f(2\cdot) \in V_{m-1}. \tag{2.18}$$

Define $W_i = V_{i+1} \backslash V_i$. Then $V_{i+1} = V_i \bigoplus W_i$. A function $f \in L^2(\Re)$ can be approximated arbitrarily well by projections, $P_i f$, into the subspaces V_i (from equation 2.17). Also, by equation 2.16, as $i \rightarrow -\infty$, the energy of the projections $P_i f$ becomes arbitrarily small. Now, define

$$\phi_{j,k}(x) = 2^{\frac{j}{2}} \phi\left(2^j x - k\right). \tag{2.19}$$

Assume the subspace V_0 is generated by a single function $\phi \in L^2(\Re)$ in the sense that $\{\phi_{0,k}, k \in \mathcal{Z}\}$ is a Riesz basis of V_0. For a set $\{\psi_k\}$ to be a Reisz (unconditional) basis, there must exist $A, B \in \Re, 0 < A \leq B < \infty$ such that

$$A\| \{c_k\} \|^2 \leq \left\| \sum_{k=-\infty}^{\infty} c_k \psi_k \right\|^2 \leq B\| \{c_k\} \|^2 \tag{2.20}$$

for all bi-infinite sequences $\{c_k\} \in l_2$. Then, all the subspaces V_j are generated by the same ϕ in the sense that $\{\phi_{j,k}, k \in \mathcal{Z}\}$ is a basis of V_j (by equation 2.18). Now, as $j \rightarrow -\infty$, the Reisz basis functions $\{\phi_{j,k}, k \in \mathcal{Z}\}$ have decreasing frequency. Therefore, the detail removed by moving from V_j to V_{j-1} must be stored in the complementary subspace W_j. Thus, the multiresolution analysis model of the wavelet transform decomposition is directly interchangeable with the high-pass/lowpass model shown in section 2.4.2.

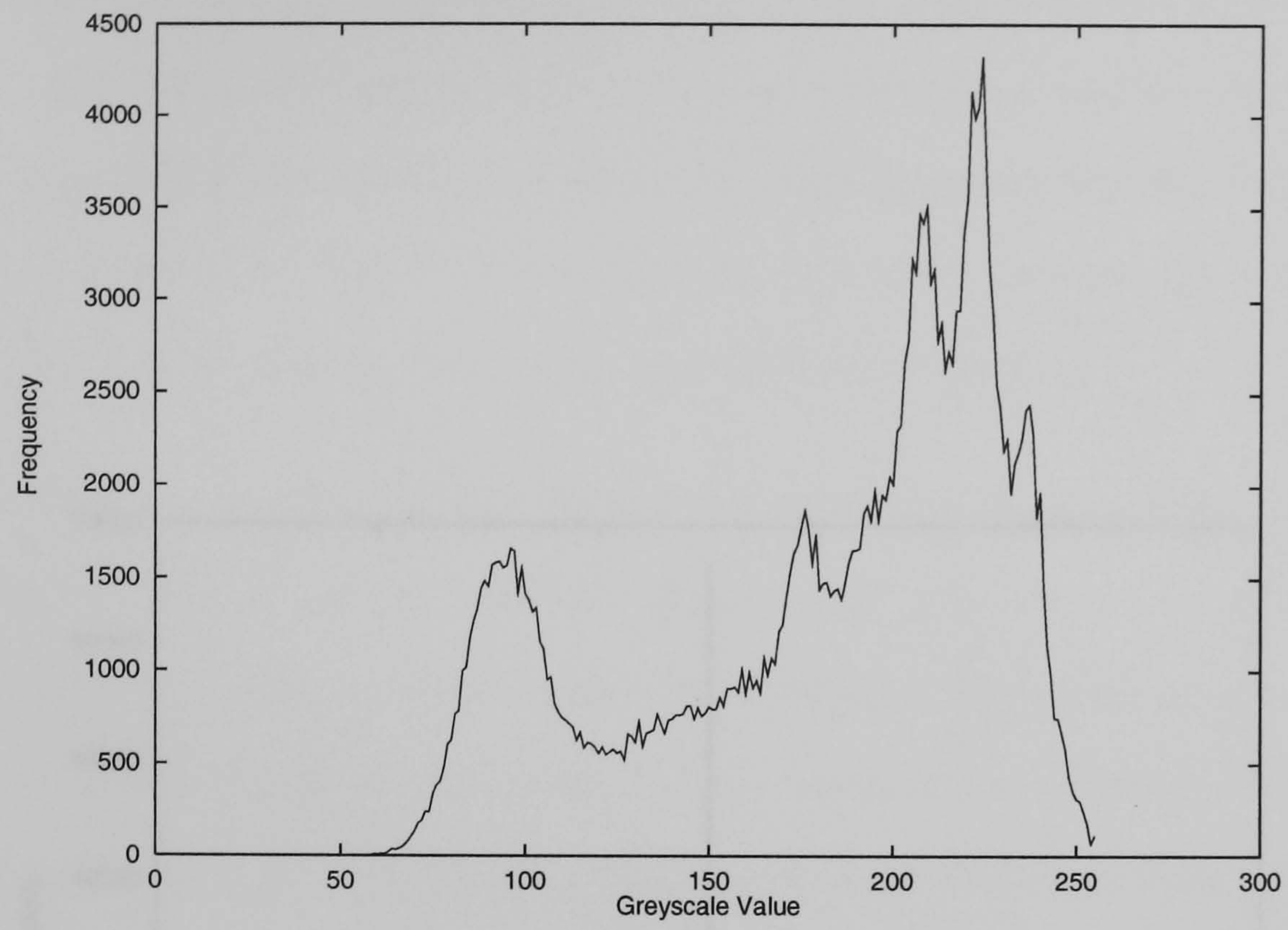


Figure 2.7: Histogram of raw data of Lena image

2.5 Properties of the Wavelet Transform

The wavelet transform is a decorrelating transform. That is, it reduces the correlation between pixels within a neighbourhood, thereby paving the way for more efficient compression of the resultant data. Figures 2.7 and 2.8 show the effect of a wavelet transform on an image histogram. Figure 2.7 shows the original histogram of the Lena 512×512 pixel image and figure 2.8 shows the histogram of the wavelet transform coefficients. The energy compaction properties of the wavelet transform are apparent from this figure.

Most, if not all, transform coding methods employ a decorrelating transform, and

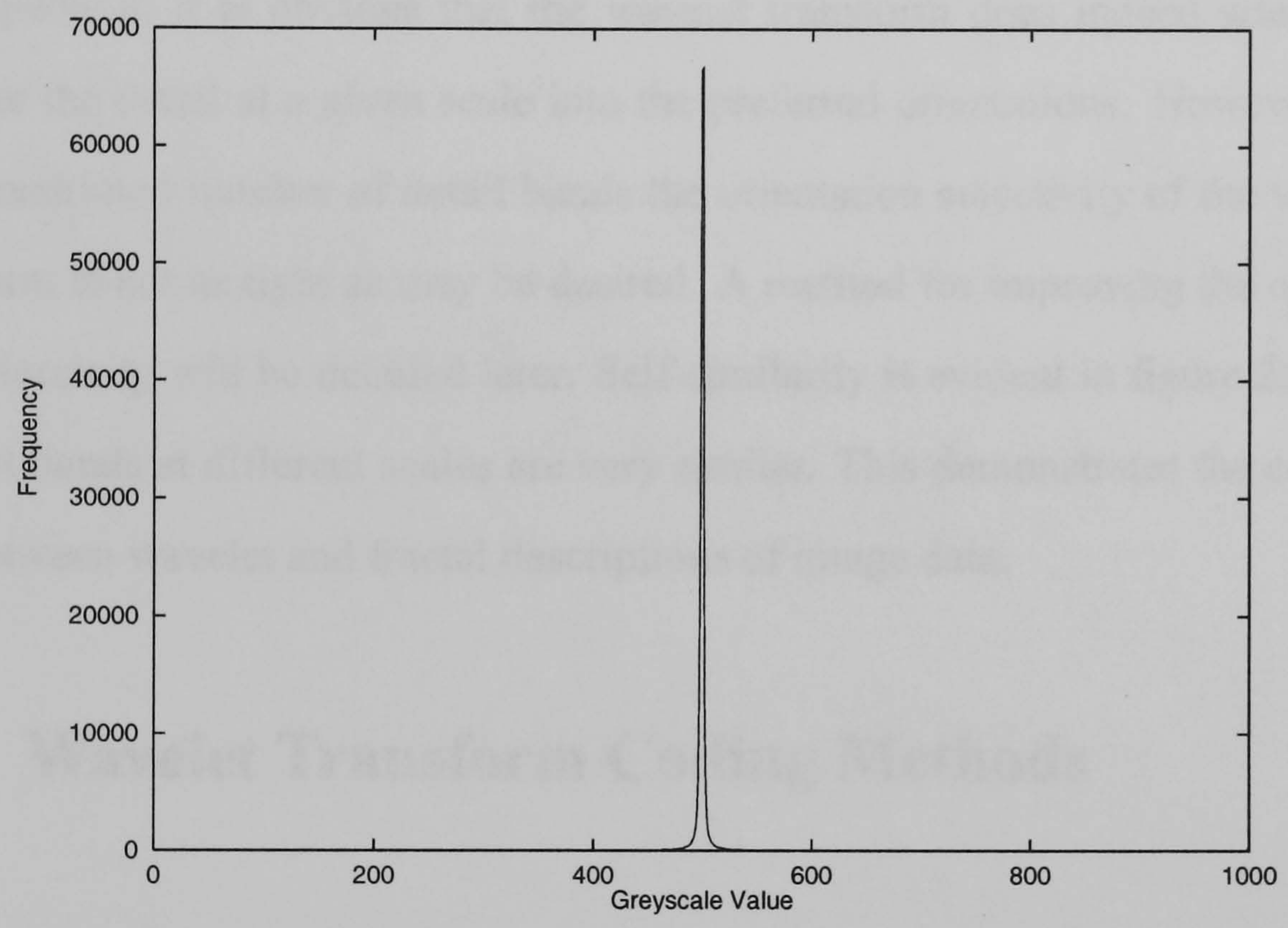


Figure 2.8: Histogram of wavelet transform coefficients of Lena image decomposed to 5 levels

the wavelet transform is very similar in this respect. What the wavelet transform does, in addition to the decorrelation of the data, is to give a multiresolution representation of the data in the horizontal, vertical and diagonal directions (the *preferred orientations*). Figure 2.9 shows a typical wavelet decomposition to three levels, shown as absolute value to give a better view of the data.

By inspection, it is obvious that the wavelet transform does indeed attempt to separate the detail at a given scale into the preferred orientations. However, due to the restricted number of detail bands the orientation selectivity of the wavelet transform is not as tight as may be desired. A method for improving the orientation selectivity will be detailed later. Self-similarity is evident in figure 2.9 - the wavelet bands at different scales are very similar. This demonstrates the connection between wavelet and fractal descriptions of image data.

2.6 Wavelet Transform Coding Methods

Wavelet transform coding has been prevalent in research circles since its introduction in [ABMD92]. These authors used a simple separable wavelet decomposition, and then coded the resulting coefficients using a vector quantizer with a multiresolution codebook, generated from a suitable sample of images. A multiresolution codebook is an amalgamation of many sub-codebooks. Once an image is decomposed via the wavelet transform, each directional (horizontal, vertical, diagonal) sub-band at each level has a codebook generated for it. This method,

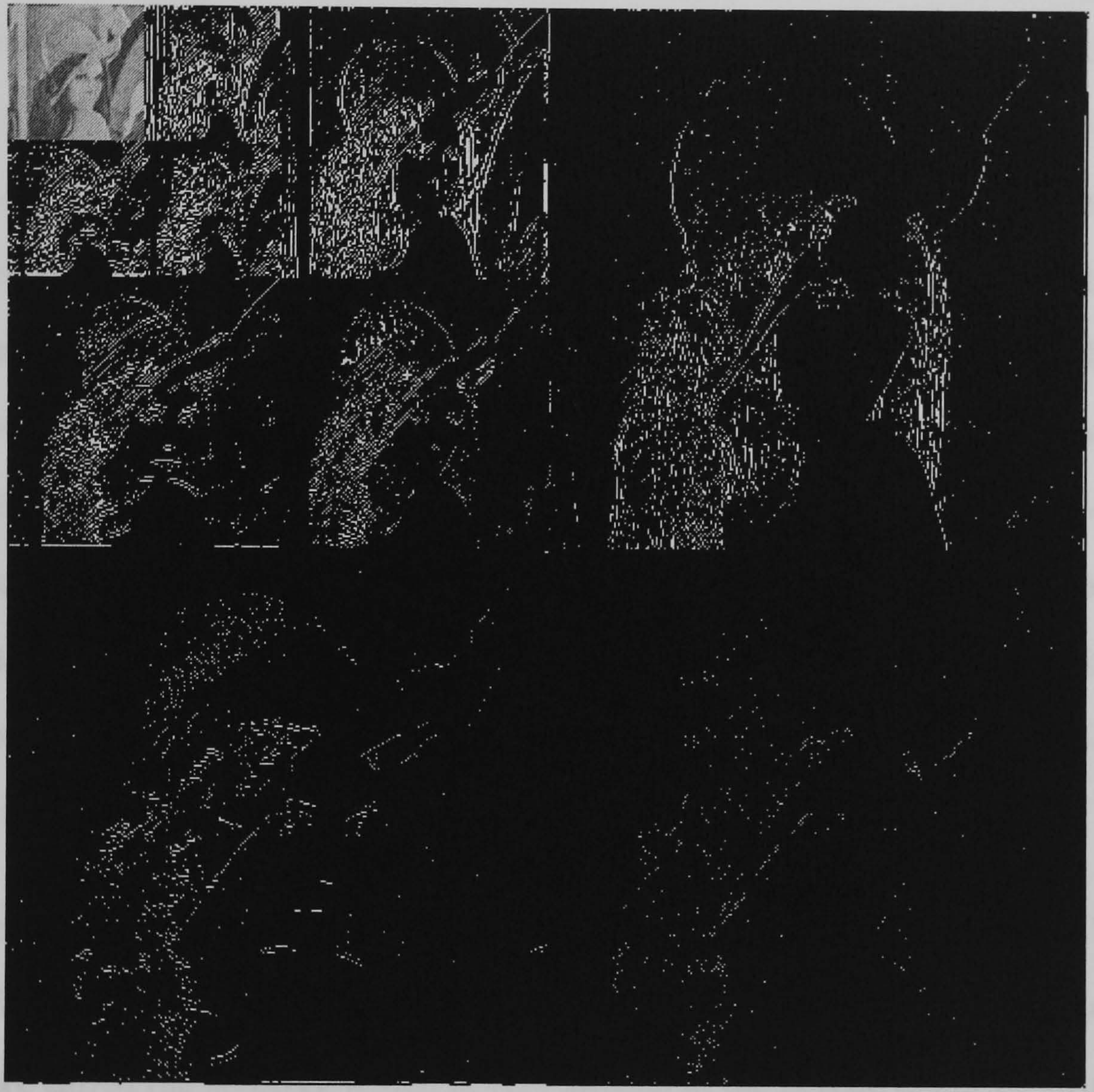


Figure 2.9: A 3-level wavelet decomposition of the Lena image using the D8 filter (absolute value)

it is argued by the authors, results in a more optimal codebook than a global one. Indeed, it is known that using a global codebook for vector quantization smooths high frequency features, such as edges and textures. Using a codebook generated on each individual wavelet sub-band results in vectors that suit the edge artifacts encountered there much better. Also, the resulting sub-codebooks are much smaller and perform better than the global codebook on their particular sub-band. The authors limit the search for each vector to be coded to the appropriate sub-codebook. That is, if a vector to be coded is taken from wavelet subband (r, d) then only the sub-codebook generated on subband (r, d) will be searched. The sub-codebooks are generated and optimized using the Linde-Buzo-Gray method, described in [LBG80]. Finally, a bit allocation scheme is employed, based on the relative visual importance of each sub-band. The authors, however, do not concentrate on obtaining the lowest possible bit rate. Instead, they provide an insight into the effects that different filter designs have on the signal-to-noise ratio of a coded image at a given bit rate.

The benchmark for wavelet transform coding algorithms remains Shapiro's embedded zerotree method [Sha93]. In this paper, Shapiro presented a new data structure called the *zerotree* which is very similar to the classic quadtree structure. Consider the coefficient tree shown in figure 2.10. The coefficient at the coarsest scale has children at each finer scale as shown in the figure, creating the quadtree structure. The zerotree is really a very simple structure, consisting of four symbols

- Zerotree root

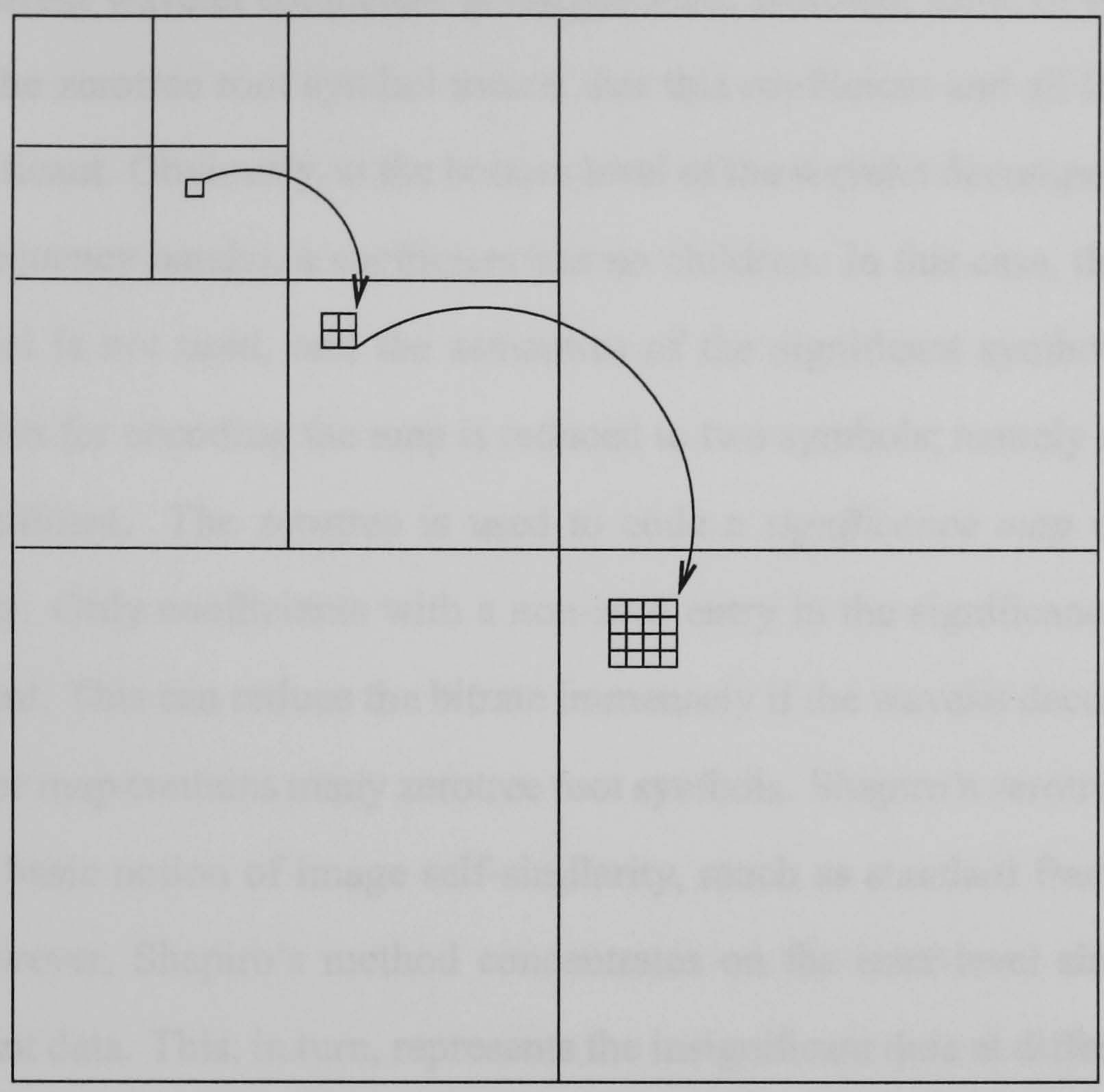


Figure 2.10: Parent-Child relationships of coefficients in a wavelet decomposition

- Isolated Zero
- Positive Significant
- Negative Significant

A threshold, T, is chosen and then each coefficient, x, is classified as *significant* if $|x| \geq T$ and *insignificant* if $|x| < T$. The positive and negative significant symbols indicate that the wavelet coefficient at that position is significant with respect to T and with the appropriate sign. The isolated zero symbol is used to encode the fact that the current wavelet coefficient is insignificant, although some of its children are not. The zerotree root symbol means that this coefficient and all its children are insignificant. Obviously, at the bottom level of the wavelet decomposition (the highest frequency bands), a coefficient has no children. In this case, the zerotree root symbol is not used, and the semantics of the significant symbols change. The alphabet for encoding the map is reduced to two symbols; namely significant and insignificant. The zerotree is used to code a *significance map* of wavelet coefficients. Only coefficients with a non-zero entry in the significance map are actually sent. This can reduce the bitrate immensely if the wavelet decomposition significance map contains many zerotree root symbols. Shapiro's zerotree encoder exploits a basic notion of image self-similarity, much as standard fractal coding does. However, Shapiro's method concentrates on the inter-level similarity of insignificant data. This, in turn, represents the insignificant data at different scales in the original image, hence the self-similarity argument. In the next chapter, the connection between wavelets and self-similarity will be exploited in the design of a still image coder.

2.7 Summary

This chapter has introduced the concept of fractal image coding using iterated functions systems. The main problem, from an image coding standpoint, is that it has been shown that it is impossible for a fractal image coder to achieve perfect reconstruction of an arbitrary signal. An analysis of the basis generated by the fractal method related the basis to that generated by the wavelet transform using quadrature mirror filters. Comparisons drawn between fractal and wavelet methods demonstrate the multiresolution nature of both. Combining the two methods into an image coding algorithm is the obvious next step.

Chapter 3

Exploiting Fractal Compression in the Wavelet Domain

Both fractal block coding and wavelet transform coding rely on the symmetries inherent in image data. As discussed in the previous chapter, the wavelet basis is a more natural basis for the fractal method than the one generated by dyadic downsampling and the addition of constant blocks. It seems, therefore, natural to attempt to unify the two disparate strategies.

3.1 A predictive wavelet transform coder

The method presented here can be seen as a predictive coder, operating in the wavelet domain. Blocks of data representing larger scales are used to predict those at smaller scales [Tod89]. The method is a recursive one, performed one wavelet level at a time, in which the preferred orientation bands are treated together.

3.1.1 Algorithm Description

The image to be coded is decomposed by the wavelet transform to a given level, say l, known as the *base* level. The first level to be approximated, called the *domain level*, is then $d < l$. Then, the *range level* is the level from which the domain level shall be approximated, that is $r = d+1$. The construction of the relevant levels is shown in Figure 3.1, where $l = 5, d = 2, r = 3$. Levels from l through r are quantized using a linear quantizer for the highpass coefficients. A separate quantizer is used for the lowpass coefficients at the highest level, since the statistics of the data are sufficiently different. In the spirit of equation 2.13, the quantized approximation is used as the range level of coefficients to reduce errors at the decoder. Each of the bands in the domain level is partitioned into non-overlapping domain blocks. Let $\{D_i^{HL}\}$, $\{D_i^{LH}\}$ and $\{D_i^{HH}\}$ be the domain blocks from each subband. For each domain block set $\{D_i^{HL}, D_i^{LH}, D_i^{HH}\}$, a corresponding range block set $\{R_i^{HL}, R_i^{LH}, R_i^{HH}\}$ must be found which minimizes the error

$$\epsilon = ||D_i^{HL} - \hat{D}_i^{HL}||^2 + ||D_i^{LH} - \hat{D}_i^{LH}||^2 + ||D_i^{HH} - \hat{D}_i^{HH}||^2 \tag{3.1}$$

where

$$\hat{D}_i^X = \iota_i(\alpha_i \cdot \hat{R}_i^X) \tag{3.2}$$

$\hat{R}_i$ is a normalised range block,

$$\alpha_i = \frac{< \hat{R}_i^{HH}, D_i^{HH} > + < \hat{R}_i^{HL}, D_i^{HL} > + < \hat{R}_i^{LH}, D_i^{LH} >}{< \hat{R}_i^{HH}, \hat{R}_i^{HH} > + < \hat{R}_i^{HL}, \hat{R}_i^{HL} > + < \hat{R}_i^{LH}, \hat{R}_i^{LH} >} \tag{3.3}$$

and $\{\iota_i\}$ is the dihedral group of the square, i.e. combinations of rotations by $\frac{\pi}{2}$ and flips around the mid-point axes. Hence, the blocks from the three subbands

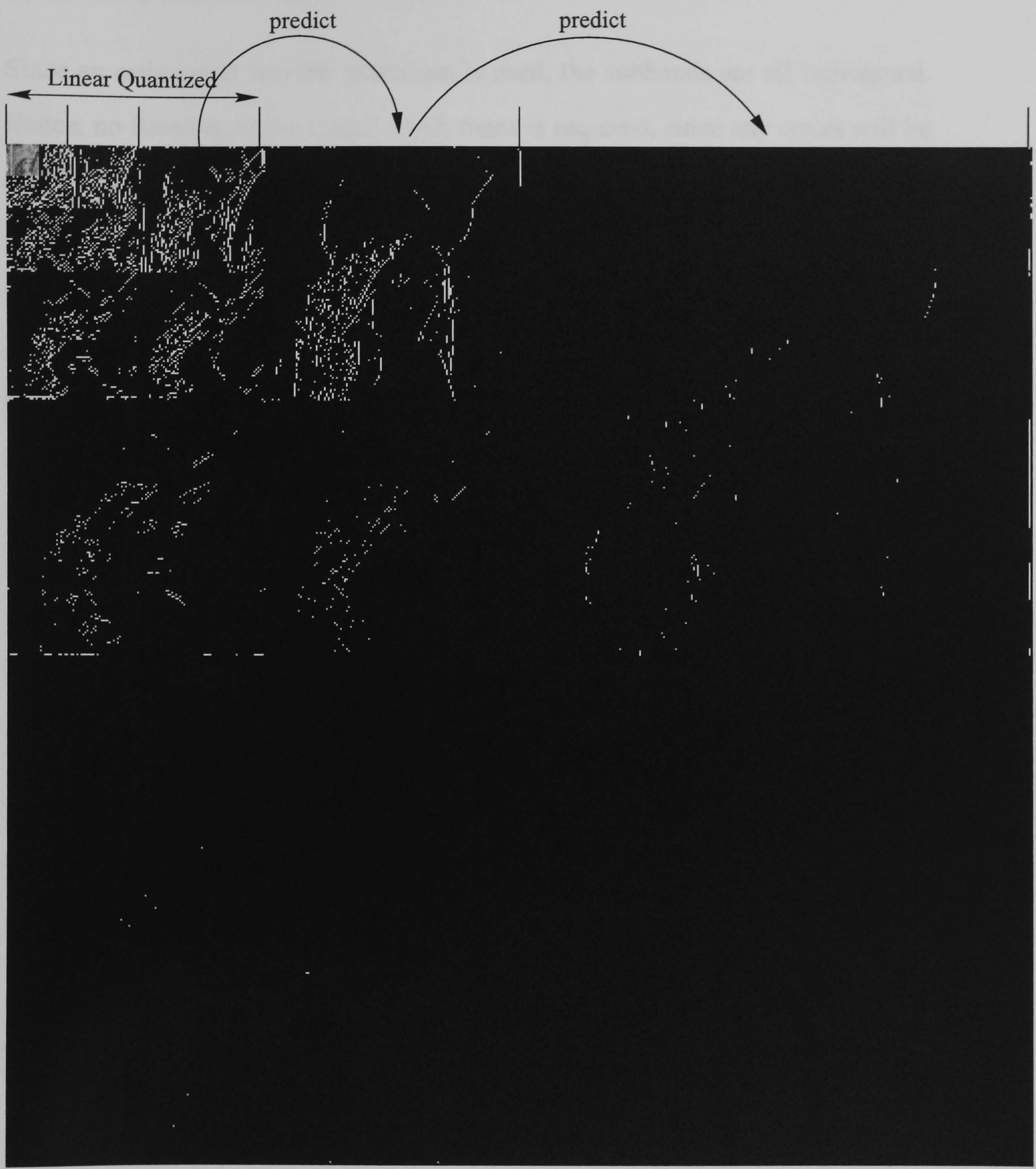


Figure 3.1: Construction for the fractal wavelet coder

are scaled by the same factor.

Since an orthogonal wavelet transform is used, the subbands are all orthogonal. Hence, no iteration of the fractal block maps is required, since any errors will be orthogonal to the approximation. Since there will be no iteration of the maps in the decoding process, the value of α_i is unconstrained. Given a domain block size s_d and search radius δ_s, define the *search region* as the subset of the subband $\mathcal{I}$ given by $([x - \delta_s s_d, x + \delta_s s_d] \times [y - \delta_s s_d, y + \delta_s s_d]) \cap \mathcal{I}$. A spiral search from the centre of the search region is used to address the range blocks. Obviously, a method which performs no searching is also possible. This method constrains the maps so that each range block R_i^X ($X \in \{HH, HL, LH\}$) is in the same relative position, but in different subbands. The reasoning behind this is that if two blocks are similar, then their horizontal, vertical and diagonal components must also be similar.

This process is then repeated on successive levels of the wavelet decomposition. The approximation of the domain level generated by the coder thus becomes the range coefficients for the next level. Separate affine map coefficients are generated between levels, up to level 1.

3.1.2 Handling Prediction Errors

The coder attempts to extrapolate a block in the range level to a block in the domain level in each subband. However, if a range block which provides an adequate approximation cannot be found, the block error will be large. To overcome this, block projection is performed after the inter-level prediction to reduce errors, in the same vein as Huang and Gharavi-Alkhansari [GAH94]. This method, however, is simpler, since it is based on the Karhunen-Loève transform. During a training phase, each error block arising from the prediction across levels is stored, along with all its symmetries. The process is repeated across a range of images and the symmetrized error vectors are accumulated. The Karhunen-Loève transform is then applied to the symmetrized error vectors produced during the prediction phases, producing an orthonormal basis of the error population. Since the population contains symmetrized vectors, the resultant basis vectors are all symmetric. Prediction errors are now handled simply by projecting any errors onto the orthonormal basis generated from the Karhunen-Loève transform.

3.1.3 The Karhunen-Loève transform

Consider a population of random vectors of the form

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{3.4}$$

Defining $m_x = \mathcal{E}\,(x)$ as the mean of x and the covariance matrix

$$C_x = \mathcal{E}\left\{(x - m_x)\,(x - m_x)^T\right\} \tag{3.5}$$

where T denotes vector transposition and $\mathcal{E}\,(\cdot)$ denotes expectation. For M vectors taken from a random population, the mean vector and covariance matrix can be approximated by the sample averages

$$m_x = \frac{1}{M}\sum_{k=1}^{M} x_k \tag{3.6}$$

and

$$C_x = \frac{1}{M}\sum_{k=1}^{M} x_k x_k^T - m_x m_x^T \tag{3.7}$$

Now, C_x is a real, symmetric $n \times n$ matrix. As such, it is guaranteed that a set of n orthonormal eigenvectors is always calculable [Ner69]. Order the eigenvalues of C_x, $\{\lambda_i\}$, in decreasing value and define row i of a matrix, A, to be the eigenvector associated with eigenvalue λ_i. Then, the Karhunen-Loève transform is defined ([GW92]) as

$$y = A(x - m_x) \tag{3.8}$$

The vectors y are then uncorrelated and have zero mean. However, for this application, only the eigenvectors of the covariance matrix, C_x are required, since they form an orthonormal basis of the underlying population. Threshold coding using a 4×4 KLT is used to code the prediction errors.

3.1.4 Entropy Coding

The parameter sets that describe the blockwise maps and the basis projection coefficients are linearly quantized and entropy coded. A *rate value* determines how many bins the quantizer uses and how many symbols the arithmetic coder can encode. Figure 3.2 shows a block diagram of the hybrid coder.

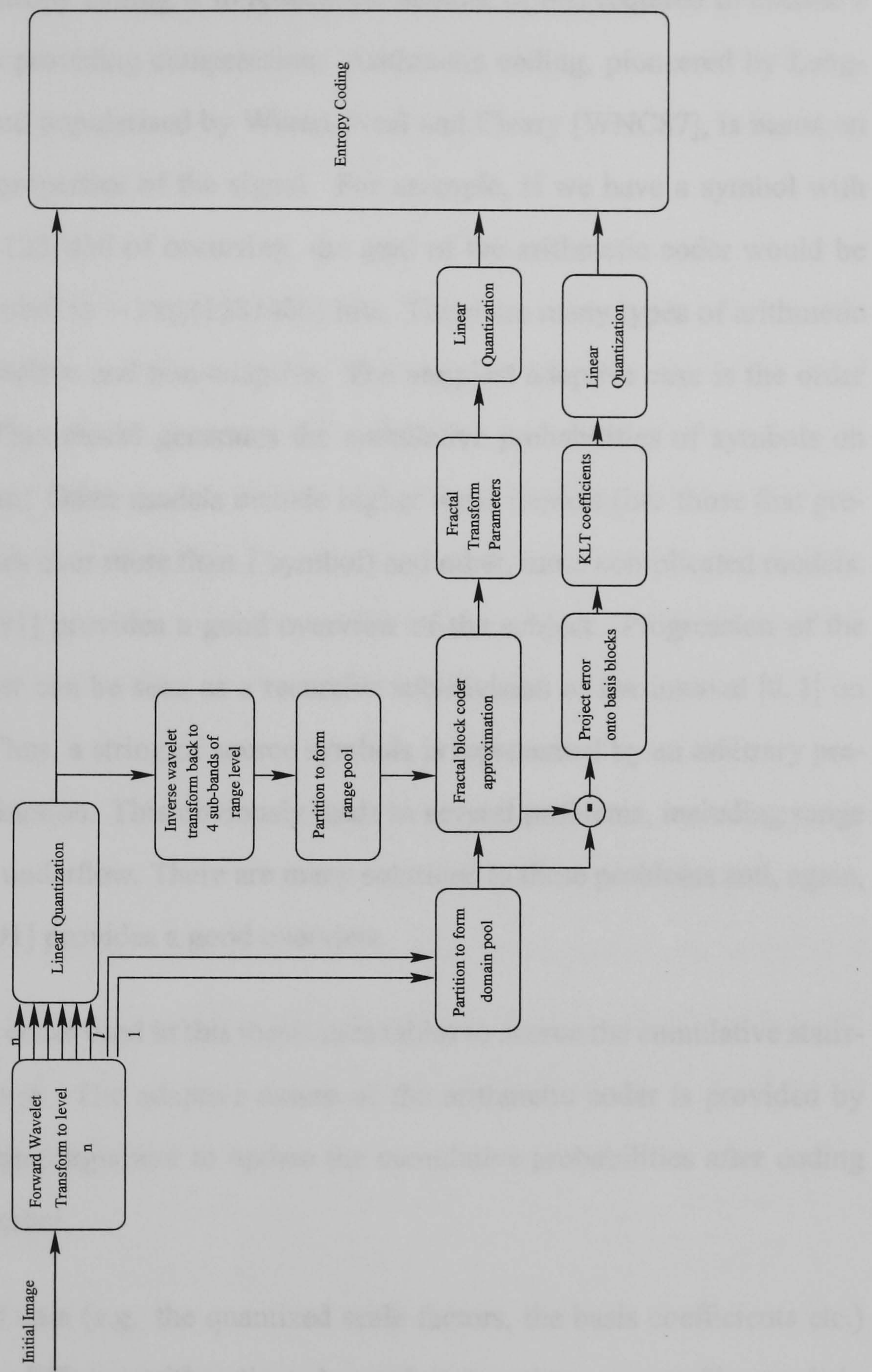


Figure 3.2: Block diagram of the hybrid image coder

The goal of entropy coding is to reduce the number of bits required to encode a symbol, hence providing compression. Arithmetic coding, pioneered by Langdon [Lan84] and popularised by Witten, Neal and Cleary [WNC87], is based on the statistical properties of the signal. For example, if we have a symbol with probability of $123/456$ of occurring, the goal of the arithmetic coder would be to code the symbol in $-\log_2(123/456)$ bits. There are many types of arithmetic coder, both adaptive and non-adaptive. The simplest adaptive case is the order zero model. This model generates the cumulative probabilities of symbols on the input stream. Other models include higher order models (i.e. those that predict probabilities over more than 1 symbol) and other, more complicated models. Williams [Wil91] provides a good overview of the subject. Progression of the arithmetic coder can be seen as a recursive sub-division of the interval $[0, 1]$ on the real line. Thus, a string of source symbols is represented by an arbitrary precision binary fraction. This obviously leads to several problems, including range resolution and underflow. There are many solutions to these problems and, again, Williams [Wil91] provides a good overview.

The arithmetic coder used in this thesis uses tables to accrue the cumulative statistics of the source. The adaptive nature of the arithmetic coder is provided by a simple counting argument to update the cumulative probabilities after coding each source symbol.

Each stream of data (e.g. the quantized scale factors, the basis coefficients etc.) is encoded by a different arithmetic coder, and streamed to separate files on disk.

This allows the arithmetic coders more accurately to model the statistics of each stream and allows for simple analysis of the resultant files.

3.1.5 Reconstruction

Reconstruction proceeds much the same as for conventional fractal coding. The lowpass and base level of the wavelet coefficients are decoded and the first set of fractal block maps is then applied to the base level. Any basis projection coefficients are applied block-wise. The reconstructed level is then used as the domain for the next set of affine transforms. Note that, unlike conventional fractal compression, the fractal block maps are not iterated. An inverse wavelet transform from the base level reconstructs the original image.

3.2 Orientation in Wavelet Subbands

Assume that the wavelet decomposition has been performed on an image to a specific level. Label the transform coefficient bands f_{LL},f_{LH},f_{HL} and f_{HH} at that level. Then the orientation of the pixel at position (x, y) in the LL band of the level below is approximated by

$$O(x,y) = \arctan\left(\frac{2 \times \left(f_{HL}(x,y) + \frac{f_{HH}(x,y)}{2}\right) \times \left(f_{LH}(x,y) + \frac{f_{HH}(x,y)}{2}\right)}{\left(f_{HL}(x,y) + \frac{f_{HH}(x,y)}{2}\right)^2 - \left(f_{LH}(x,y) + \frac{f_{HH}(x,y)}{2}\right)^2}\right). \tag{3.9}$$

The orientation estimate can be used further to reduce the size of the pool to be searched. Since a set of isometries will be applied to the blocks, the orientation estimates are normalized into a canonical orientation, that lying in the interval $[0, \frac{\pi}{2}]$. The orientation of a block is then defined as the average of the orientations of the pixels within the block. Now, the orientation estimates are linearly quantized. If the orientation block map of the range level is precomputed before any prediction begins, the overhead of the orientation calculation for each domain block becomes negligible in the light of the error calculation required for each range block tested. The coder is modified so that the spiral search does not include blocks whose pre-computed orientation does not match that of the quantized domain block orientation. Furthermore, by not including these blocks, the distribution of search termination values will be compacted, thereby increasing efficacy of the entropy coder. Figure 3.3 and Figure 3.4 show the effect of not including blocks whose orientations do not match.

3.3 Rate Control

Rate control is provided in a variety of ways. The granularity of the quantizer used on the scale factor α_i is the most obvious parameter for variation. The quantizer bin size used for the wavelet lowpass coefficients and the linear quantized levels also have a large impact on bit rate and reconstruction quality. The basis projection coefficients are also quantized and this quantizer granularity can again be varied to adjust overall bit rate and quality. Finally, the granularity of the orientation quantizer and the radius of the block search can be varied. In the following coding

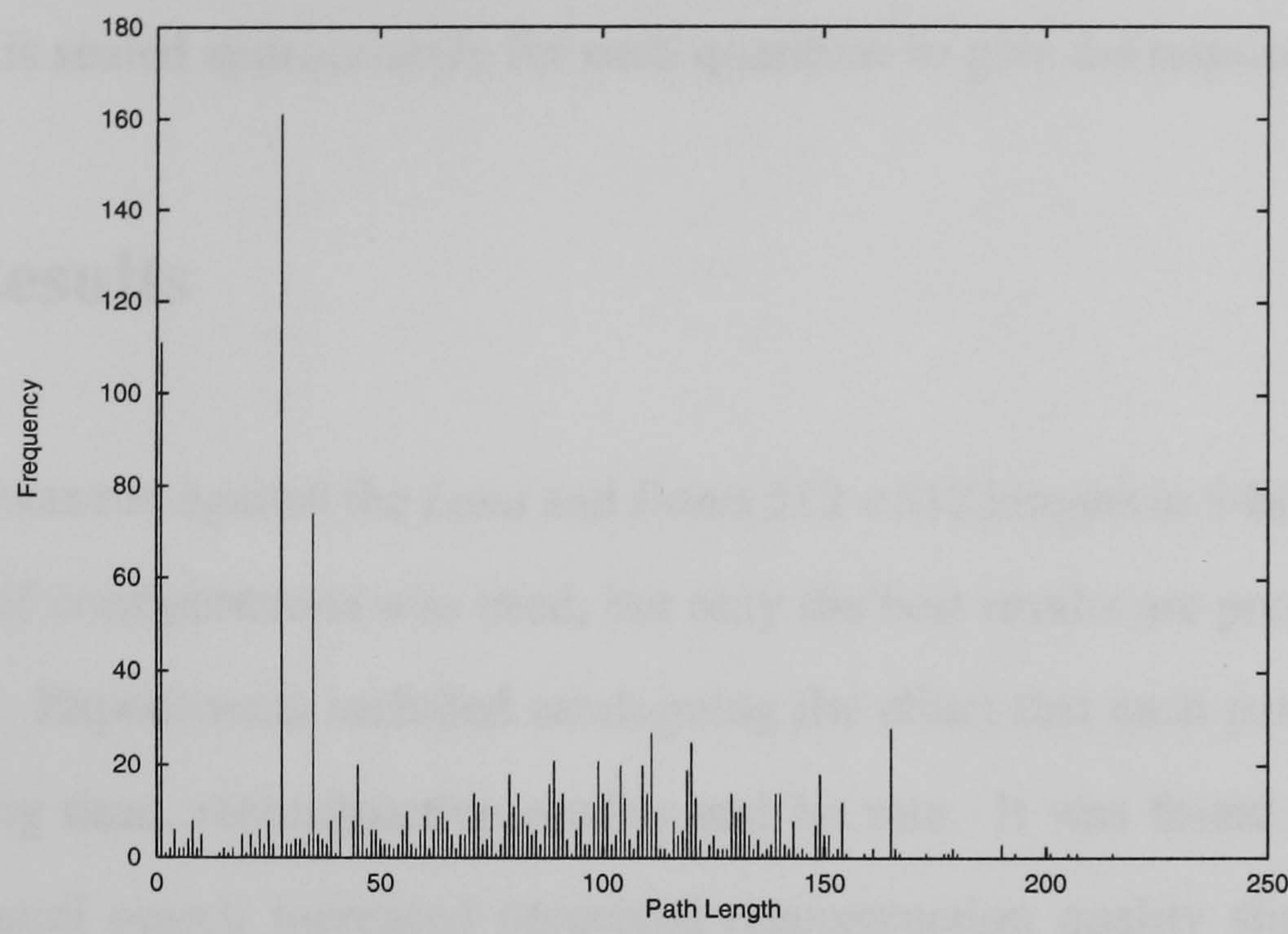


Figure 3.3: Distribution of path positions when block orientations are not matched

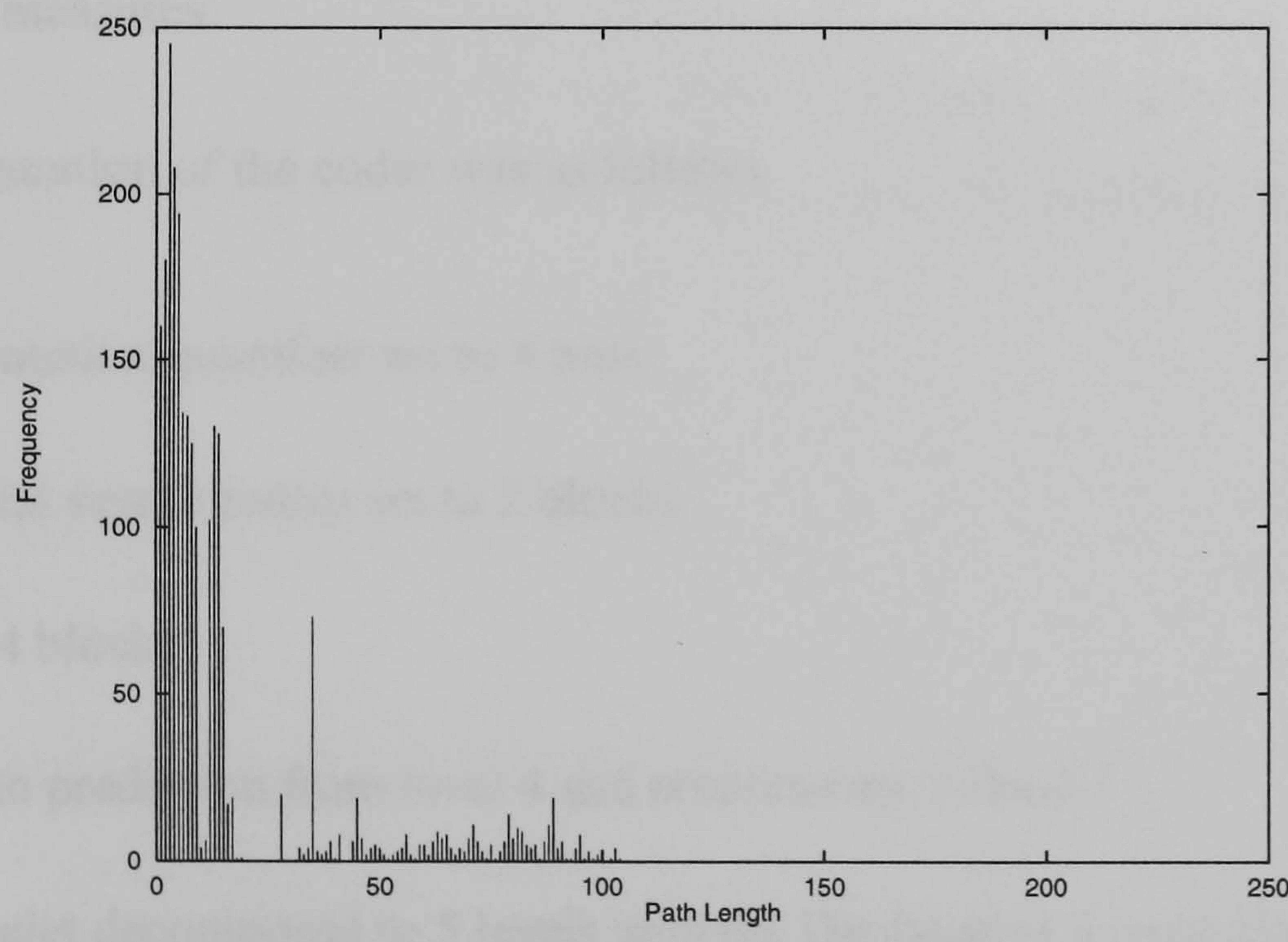


Figure 3.4: Distribution of path positions when block orientations are matched

runs, the α_i and basis quantizers are controlled by a single rate parameter. The rate parameter is scaled appropriately for each quantizer to give the required bin size.

3.4 Results

The coder was run against the *Lena* and *Boats* 512×512 images in 8-bit greyscale. A variety of configurations was tried, but only the best results are presented here for brevity. Experiments included cataloguing the effect that each parameter had on encoding time, reconstruction quality and bit rate. It was found that an extensive spatial search increased perceived reconstruction quality slightly while increasing encoding time inordinately. The wavelet filter also had an effect on the reconstruction, although this was mainly in the subjective quality as opposed to numerical measures.

The configuration of the coder was as follows

- Orientation quantizer set to 4 bins
- Spatial search radius set to 2 blocks
- 4×4 blocks
- Begin prediction from level 4 and continue up to level 1
- Wavelet decomposed to 5 levels with the Daubechies 8 point filter [Dau88]
- Low frequency wavelet quantizer rate value set to 64

Image	*Rate Value*	*Rate (bpp)*	*PSNR*
Lena	64	2.62	39.80
Lena	128	1.64	38.28
Lena	256	0.92	36.18
Lena	512	0.53	34.04
Lena	1024	0.32	31.52
Lena	2048	0.20	29.06
Lena	4096	0.14	27.12
Boats	64	2.80	38.78
Boats	128	1.81	37.58
Boats	256	1.12	35.63
Boats	512	0.69	32.94
Boats	1024	0.43	29.94
Boats	2048	0.27	27.17
Boats	4096	0.18	25.17

Table 3.1: Results for Predictive Wavelet Coder

- Orientation-matched search

Table 3.1 shows the results for the simlations performed. Figure 3.5 shows the rate distortion curves for the *Lena* and *Boats* images. Reconstructions at various bit rates are shown in figures 3.6 and 3.7.

3.5 Conclusions

The coder presented in this chapter appears to work well at high to medium bit rates, in the range $3bpp$ to $0.25bpp$. However, at lower bit rates, the artifacts become visually disturbing, for example as in figure 3.6(d) and 3.7(d). Orientation has been shown to be a good measure of the type of similarity between image

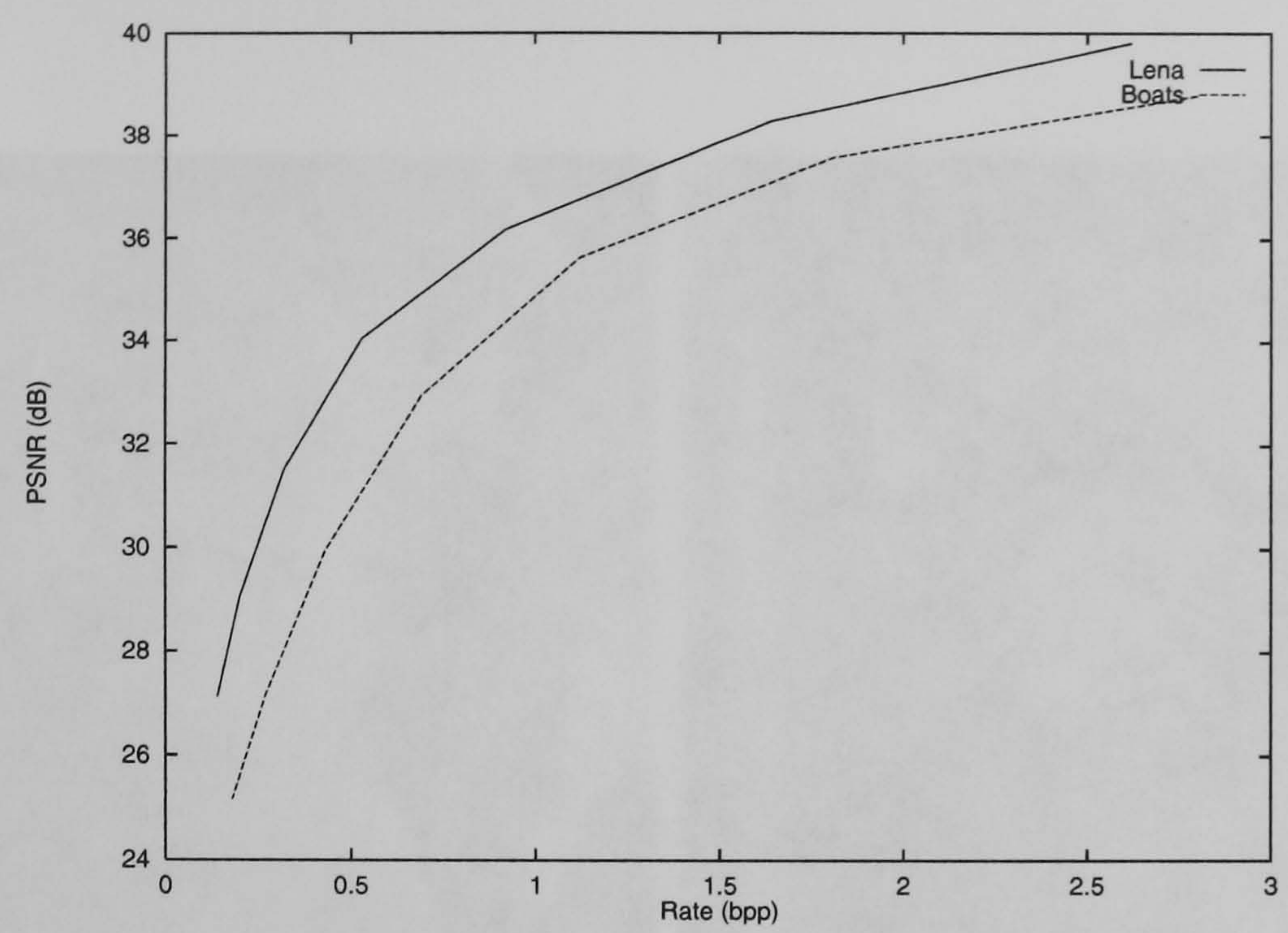


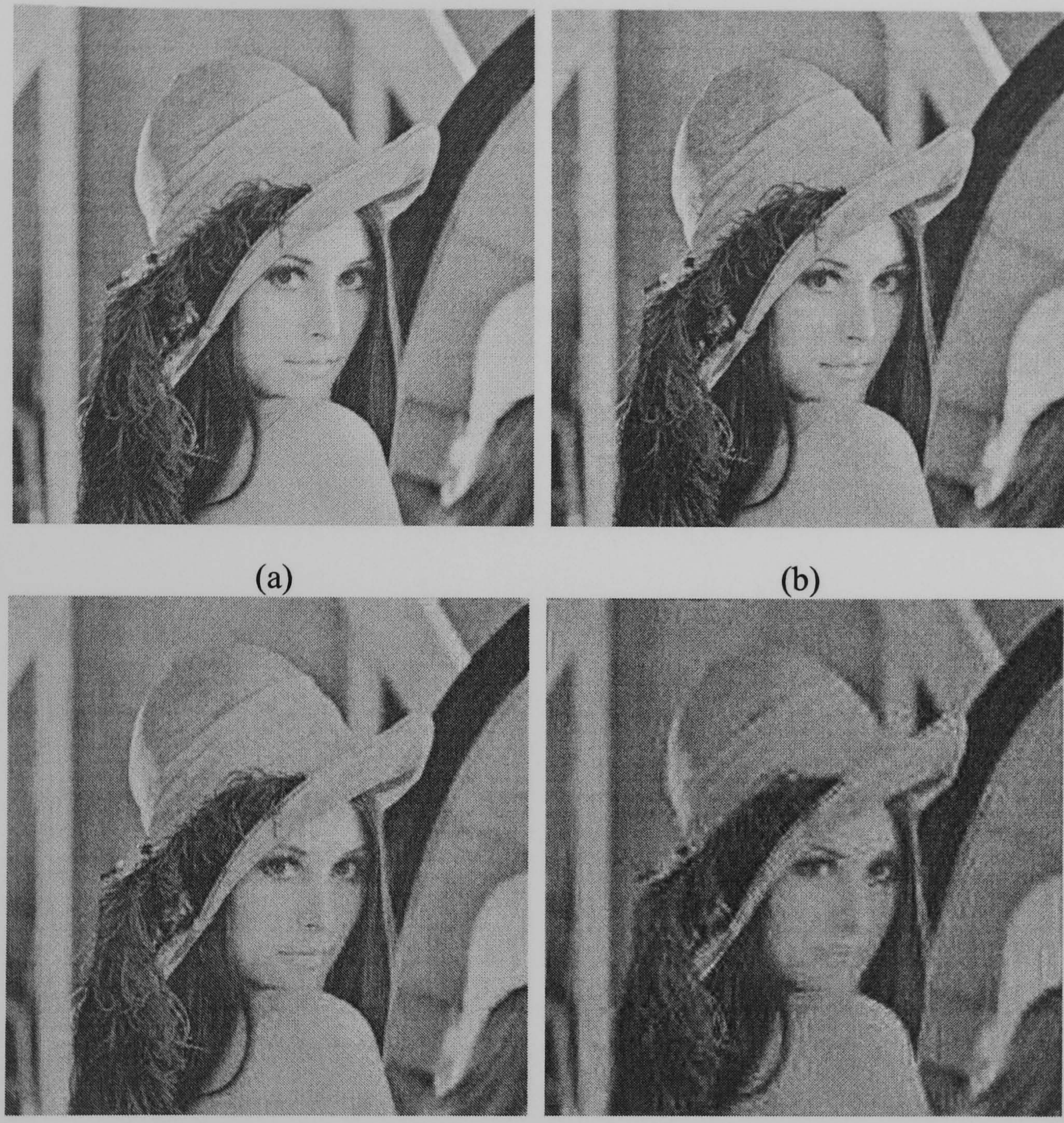
Figure 3.5: Rate-Distortion Curves for Predictive Wavelet Coder

patches [Tod89]. The link between fractal image coding and the wavelet basis has been made concrete by this image coding algorithm.

However, the system does suffer from problems. In particular, the system is still asymmetric; that is, the encoding time is disproprtionately longer than the decoding time. Encoding times for the results presented here are of the order of 15 minutes, whilst decoding times are approximately 20 seconds [1] [2]. Furthermore, the pool of blocks to predict from is severely limited by the spatial restriction on the search. It would be simple to construct an image for which the coder could not find a good approximation to a block within the specified spatial search region.

[1] Based on execution upon a Sun Ultra 1, 143 MHz using no code optimizations. The machine was not fully quiescent while the test were being performed.

[2] Times given do not include the actual loading or saving of the image data.



(a) (b)

(c) (d)

Figure 3.6: Reconstructions of *Lena* image, coder configuration 1. (a) Original, (b) $Rate = 64, 2.62bpp, 39.80dB$, (c) $Rate = 1024, 0.32bpp, 31.52dB$, (d) $Rate = 4096, 0.14bpp, 27.12dB$

(a)

(b)

(c)

(d)

Figure 3.7: Reconstructions of *Boats* image, coder configuration 1. (a) Original, (b) $Rate = 64, 2.80bpp, 38.78dB$, (c) $Rate = 1024, 0.43bpp, 29.94dB$, (d) $Rate = 4096, 0.18bpp, 25.17dB$

In this case, the coder would degenerate to a simple basis projection and would obviously perform poorly.

Chapter 4

Predictive Wavelet Image Coding Using an Oriented Transform

As mentioned in section 2.5, the wavelet transform attempts to group image data into bands of oriented features at each scale. From the previous chapter, orientation is important in still image coding applications. The frequency domain tessellation in figure 4.1 shows the split into frequency bands provided by the conventional wavelet transform. The predictive coding scheme presented in chapter 3 requires a search of some sort in order to find the best block from which to predict. This search could be restricted without adversely affecting reconstruction quality by limiting the search to *similar* blocks. The metric of similarity could be arbitrarily defined, but grouping the blocks by orientation is a natural option. However, the conventional two dimensional wavelet transform does not provide sufficient resolution in orientation for this to be feasible.

4.1 An Oriented Wavelet Transform

In order to increase the orientation selectivity of the transform, it would be useful to increase the number of high-pass bands that the wavelet transform creates [CMW] [RV93]. This would, in turn, offer greater orientation selectivity within each band. Wilson and Spann [WS88] presented an invertible multiresolution transform which provides six orthogonal bandpass features at each scale. This was presented in the framework of image segmentation, where reconstruction is not important. Obviously, for image coding applications, perfect or near-perfect reconstruction is generally required. Treil, Mallat and Bajcsy [TMB89] present a slightly modified two dimensional dyadic wavelet decomposition in which there are four highpass bands. Consider the frequency domain tessellation shown in figure 4.1. This figure shows how the frequency spectrum is split by using QMF wavelet transform filters. The band L is the lowpass band while the bands $1, \ldots, 3$ are the highpass bands in each preferred orientation. In [TMB89] the authors split the diagonal oriented band (3) into two. The split of the band is denoted by the '!' characters. This provides a small increase in the orientation selectivity of the transform.

For the purposes of image compression, orthogonal wavelets are the preferred analysis tool since the wavelet sub-bands are then orthogonal. Obviously any orientation selectivity increasing modification should attempt to preserve as many desirable properties of the orthogonal wavelet transform as possible. The most

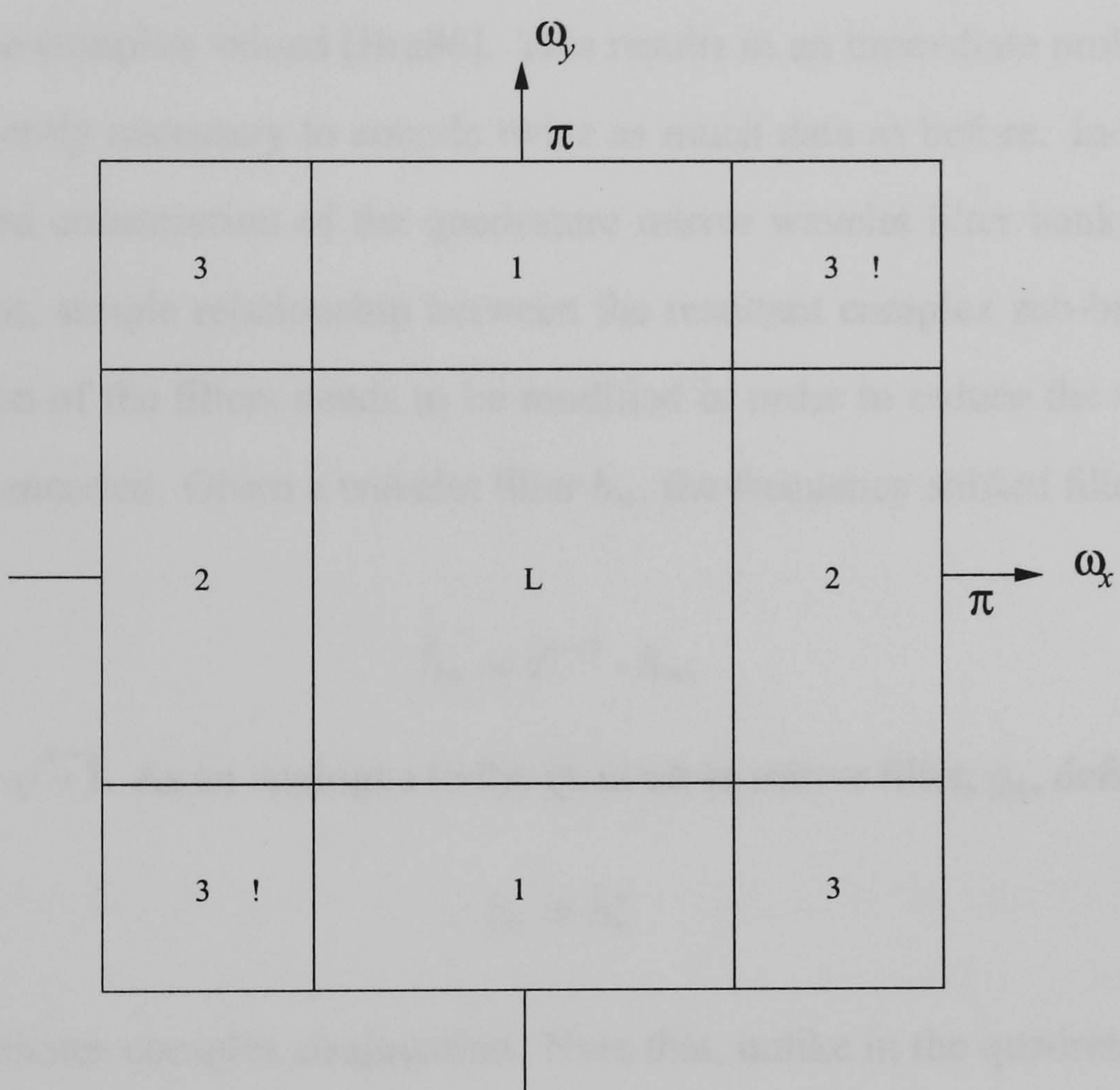


Figure 4.1: The spectrum tessellation using a conventional wavelet transform

intuitive way of making the wavelet transform more orientation selective is further to decompose the high pass detail bands, and the most natural two dimensional sub-decomposition would be into four sub-bands, since the underlying sampling is dyadic. This is simple to achieve by frequency shifting the wavelet filters by $\frac{\pi}{2}$ before applying the transform to the highpass wavelet bands. Now, the resultant data will be complex valued [Bra86]. This results in an immediate problem. It is now apparently necessary to encode twice as much data as before. Indeed, with the standard construction of the quadrature mirror wavelet filter banks, there is no apparent, simple relationship between the resultant complex sub-bands. The construction of the filters needs to be modified in order to reduce the volume of data to be encoded. Given a wavelet filter h_n, the frequency shifted filter is given by

$$\hat{h}_n = \imath^{n-1} \cdot h_n, \tag{4.1}$$

where $\imath = \sqrt{-1}$. As an analogue to the quadrature mirror filter, g_n, define

$$\hat{g}_n = \hat{h}_n^*$$

where $*$ denotes complex conjugation. Note that, unlike in the quadrature mirror filter construction, there is no time reversal of the filter. With this construction, the quadrature mirror filter property is replaced by simple hermitian symmetry [Bra86]. The application of these filters for the forward wavelet transform remains intact. The inverse transform, however, is affected since the anti-aliasing properties of the quadrature mirror filter construction no longer apply. The construction of the inverse filters will be covered later in this chapter. Applying these

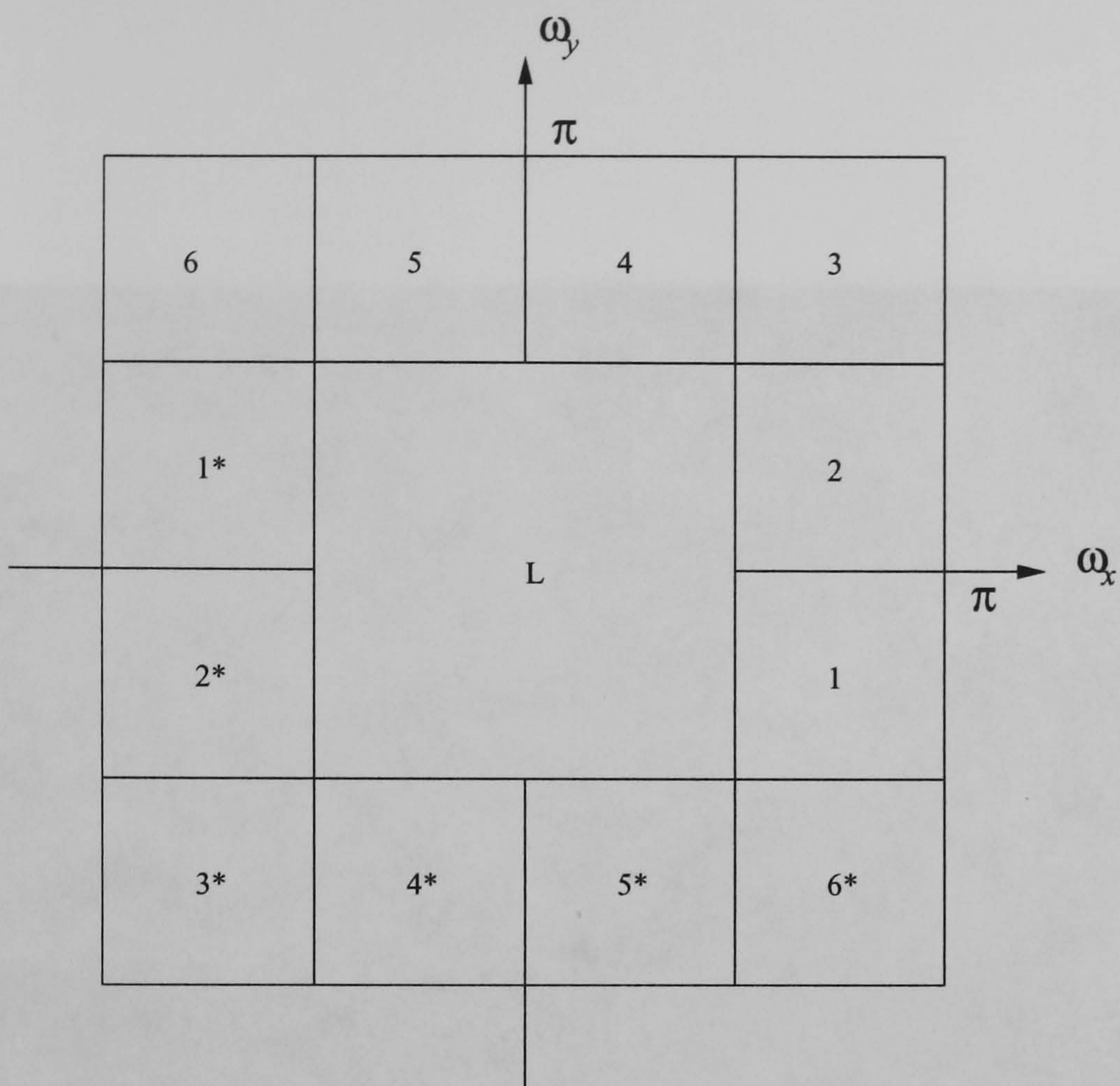


Figure 4.2: A modified spectrum tessellation

complex valued filters to the highpass bands produced by the real wavelet transform gives the frequency domain tessellation shown in figure 4.2. Here, the $*$ represents complex conjugation.

Figure 4.3 shows an FM test pattern decomposed by the complex wavelet transform. The image is decomposed by the conventional (real) transform to one level and then further decomposed by the oriented transform. The colours in figure 4.3 represent the complex phase of the values.

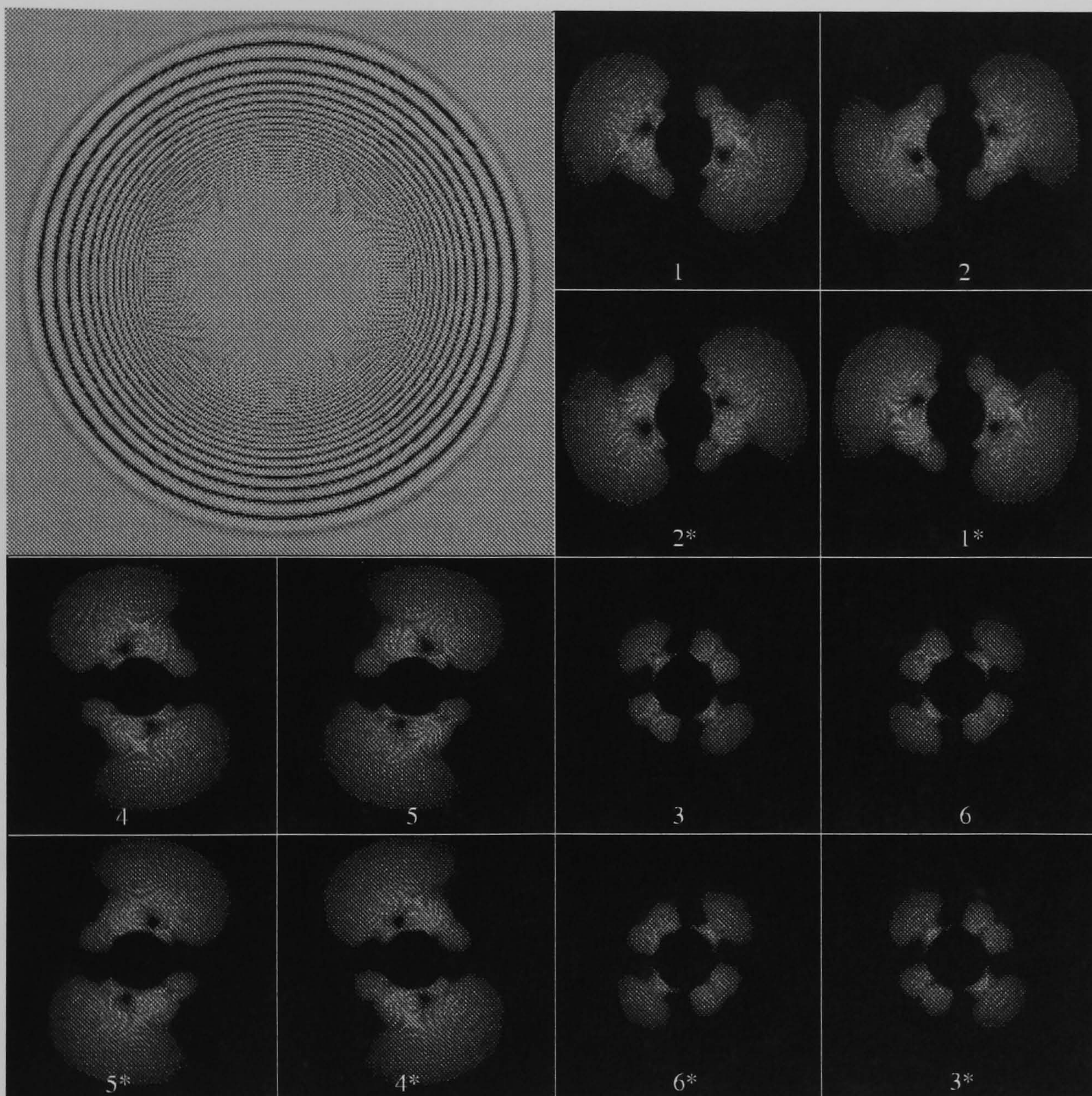


Figure 4.3: An FM test pattern decomposed by the complex wavelet transform.

Source Band	*Destination Band*	*Spatial Operation*
1	2	Reflect about X axis
2	4	Rotation by $\frac{\pi}{4}$
3	6	Rotation by $\frac{\pi}{4}$
1	5	Rotation by $\frac{\pi}{4}$

Table 4.1: Spatial relationships between complex wavelet subbands

Now, since each band represents data in a certain spatial direction, the data in one band may be related to that in another by simple spatial rotations. Table 4.1 shows the relationships between the complex subbands. Bands 1,2,4 and 5 are related by appropriate rotations and reflections as are bands 3 and 6, and there are no relationships between the two groups. These symmetry properties can be exploited in the coder design.

4.2 Filter Design Considerations

As stated in section 4.1, the conjugate filter construction does not include a time reversal. As such, any perfect reconstruction properties of the filter will be lost, since the filter pair will no longer satisfy the conjugate mirror requirement [PM96],

$$G(z) = -H(-z^{-1}).$$

In order to find the required inverse for a given filter, inversion of the transform matrix was used. The method of creating the analysis filters for the $BIORTH$ family was based on a simple iterative band and index limiting [Wil97]. These filters do not satisfy perfect reconstruction, but the unquantized reconstruction error is vanishingly small when compared to any coding errors. For reference, the filters detailed in tables 4.2, 4.3, 4.4 and 4.5 have been produced for use with the modified conjugate construction and are used throughout the remainder of this work. Note that the coefficients are presented before the filter has been shifted by $\frac{\pi}{2}$ as in equation 4.1.

Note that the filters BIORTH610, BIORTH1810 and BIORTH1022 all have symmetric analysis filters. This should improve their performance around edges and reduce ringing artifacts. However, the filter $BIORTH1810$ analysis filter is longer than the synthesis filter, which may introduce undesirable visual artifacts in reconstruction.

4.3 A Tree Structured Vector Quantizer

The block coding algorithm presented in the previous chapter is much less computationally intensive than conventional fractal compression. However, the block search is restricted in order to increase speed, reducing the pool of available vectors. Furthermore, performing a mean square error test on each block is restrictive and a simpler, less expensive method would be preferable. In effect, the domain block pool is simply a set of vectors, which has to be searched to identify the nearest one to each range block. In other words, it is vector quantization (VQ).

Building a history of previously encoded blocks should also reduce reconstruction error, since the coder is given a more richly populated vector space to predict

Coefficient Index	*Filter Coefficient*	*Inverse Coefficient*
1	0.2303778	-0.0007015
2	0.7148465	-0.0004066
3	0.6308807	0.0019766
4	-0.0279837	-0.0017954
5	-0.1870348	0.0080513
6	0.0308413	-0.0021939
7	0.0328830	-0.0291039
8	-0.0105974	0.0248194
9		0.0273771
10		0.195855
11		0.699447
12		0.653228
13		0
14		-0.216112
15		0
16		0.0714978
17		0
18		-0.0236541
19		0
20		0.0078256
21		0
22		-0.0025890
23		0
24		0.0008565
25		0
26		-0.0002833

Table 4.2: Filter coefficients for DAUB8CONJ (inverse is shifted by 9 places)

Coefficient Index	*Filter Coefficient*	*Inverse Coefficient*
1	0.0771225	-0.0058811
2	-0.0647889	-0.0083333
3	-0.6998960	0.0641735
4	-0.6998960	-0.0763900
5	-0.0647889	-0.7002490
6	0.0771225	-0.7002490
7		-0.0763900
8		0.0641735
9		-0.0083333
10		-0.0058811

Table 4.3: Filter coefficients for BIORTH610 (inverse is shifted by 2 places)

Coefficient Index	*Filter Coefficient*	*Inverse Coefficient*
1	0.0017493	-0.0022409
2	0	0.0046486
3	-0.0082112	-0.0092404
4	0	0.1684570
5	0.0384498	0.6866660
6	0.0024626	0.6866660
7	-0.1771298	0.1684570
8	0.0085573	-0.0092404
9	0.7237818	0.0046486
10	0.7237818	-0.0022409
11	0.0085573	
12	-0.1771298	
13	0.0024626	
14	0.0384498	
15	0	
16	-0.0082112	
17	0	
18	0.0017493	

Table 4.4: Filter coefficients for BIORTH1810 (inverse is shifted by 4 places)

Coefficient Index	*Filter Coefficient*	*Inverse Coefficient*
1	0.0521846	-0.0016230
2	-0.0139157	0
3	-0.113931	0.0052149
4	0.177472	0
5	0.67276	-0.0167076
6	0.67276	-0.014554
7	0.177472	0.0542312
8	-0.113931	-0.0259158
9	-0.0139157	-0.16489
10	0.0521846	0.133683
11		0.675275
12		0.675275
13		0.133683
14		-0.16489
15		-0.0259158
16		0.0542312
17		-0.014554
18		-0.0167076
19		0
20		0.0052149
21		0
22		-0.0016230

Table 4.5: Filter coefficients for BIORTH1022 (inverse is shifted by 6 places)

from. Vector Quantization is a common method of providing lossy data compression. A conventional vector quantizer works by providing a *codebook* of vectors which is used to approximate input vectors. The main problem with general VQ methods is that they are highly computationally intensive. One method of overcoming this is to structure the quantizer. A particular example of this subclass is the tree structured vector quantizer. There have been some attempts to generate Tree Structured Vector Quantizers [PC94] [BPL95]. These, however, work in an off-line manner (all the data must be present when the tree is built) or impose some condition on the source. For the present application, an incremental method is desirable. Using a tree structured vector quantizer, the pool of previously encoded blocks can be structured on-line and searched quickly. This is achieved by combining a simple on-line tree-growing algorithm with restructuring of the tree, using a modified form of the Linde-Buzo-Gray (LBG) [LBG80] algorithm.

Consider, therefore, a set of data in the Euclidean k-space, $\{x_i : i = 0, \ldots, k-1\} \in \Re^k$. Assuming $k \geq 2$, it is possible to partition the data into two clusters, C_1 and C_2, separated by a decision hyperplane, defined by

$$H = \{x_i \in \Re^n : ||x_i - \mu_1||^2 = ||x_i - \mu_2||^2\} \tag{4.2}$$

where μ_1 and μ_2 are the centroids of the clusters, as shown in Figure 4.4. The tree is based on the recursive application of this partitioning.

Expanding the condition in equation 4.2

$$||x_i - \mu_1||^2 = ||x_i - \mu_2||^2 \tag{4.3}$$

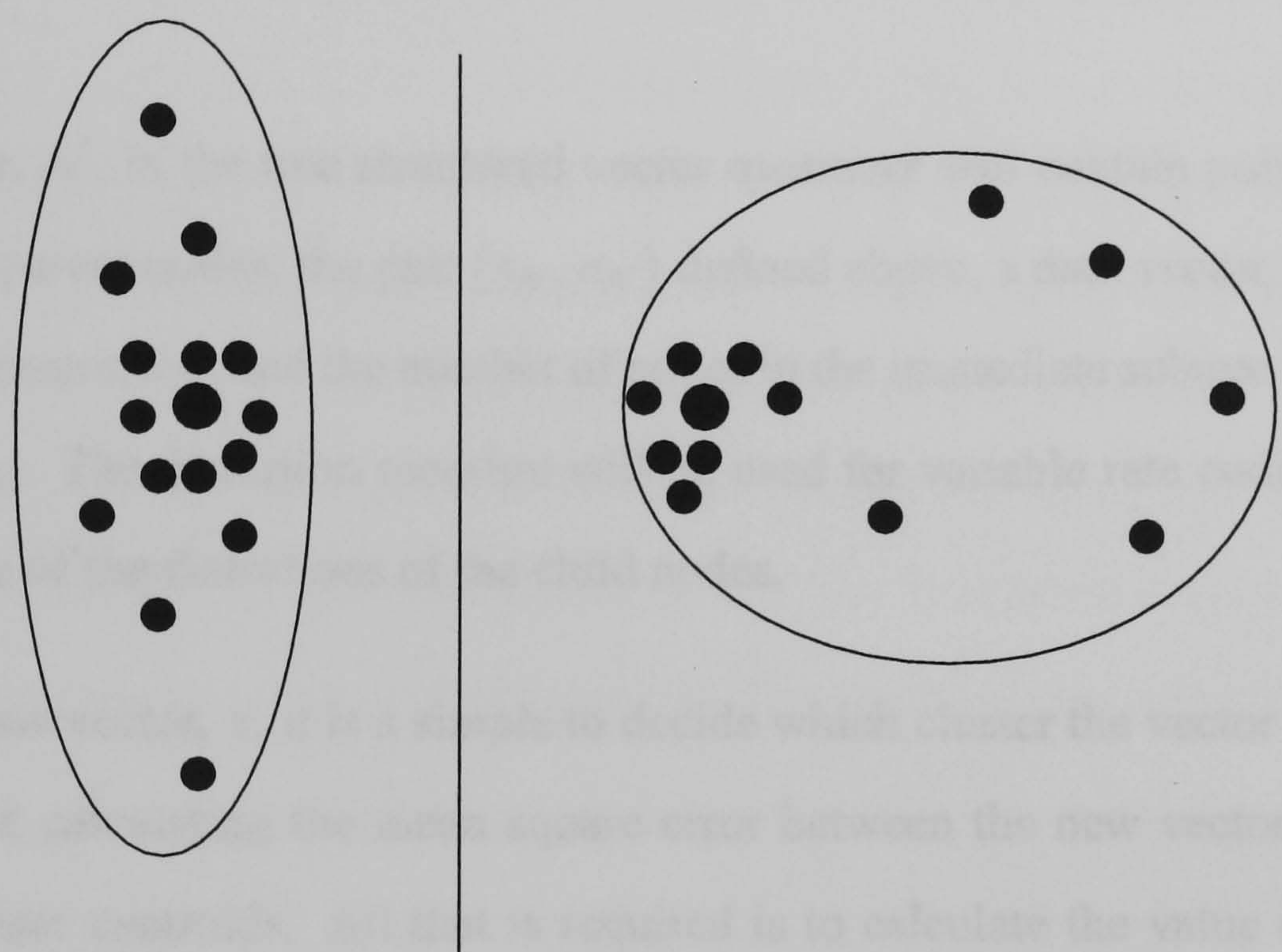


Figure 4.4: A partition of data into two clusters and the associated decision 'hyperplane' (represented by the line)

gives us

$$2x_i^T(\mu_2 - \mu_1) = \mu_2^T\mu_2 - \mu_1^T\mu_1 \tag{4.4}$$

Hence, $\mu_2 - \mu_1$ is the direction of the decision hyperplane normal and $\mu_2^T\mu_2 - \mu_1^T\mu_1$ is the distance along the normal to the point of intersection with the hyperplane itself. For ease of notation, define $z_{\mathcal{N}} = \mu_2 - \mu_1 \in \Re^k$ and $a_{\mathcal{N}} = \mu_2^T\mu_2 - \mu_1^T\mu_1 \in \Re$ at node $\mathcal{N}$.

Each node, $\mathcal{N}$, in the tree structured vector quantizer will contain pointers to its child and parent nodes, the pair $(z_{\mathcal{N}}, a_{\mathcal{N}})$ defined above, a data vector, $r_{\mathcal{N}}$, a distortion measure, $d_{\mathcal{N}}$, and the number of nodes in the immediate subtree (the usage count), $n_{\mathcal{N}}$. The distortion measure will be used for variable rate coding, and is an average of the distortions of the child nodes.

Given a new vector, x, it is a simple to decide which cluster the vector should lie in, without calculating the mean square error between the new vector and each of the cluster centroids. All that is required is to calculate the value of $2x^T z_{\mathcal{N}}$. Obviously,

$$\begin{array}{ll} 2x^T \cdot z_{\mathcal{N}} < a_{\mathcal{N}} & \text{if } x \in C_1 \\ 2x^T \cdot z_{\mathcal{N}} > a_{\mathcal{N}} & \text{if } x \in C_2 \end{array} \tag{4.5}$$

with an arbitrary choice being made if equality occurs.

All dichotomies down a particular path are evaluated until a leaf node is reached. As each node is passed, its average distortion $d_{\mathcal{N}}$ is updated by the following simple counting argument

$$d_{\mathcal{N}} = \frac{(d_{\mathcal{N}} \times n_{\mathcal{N}}) + d(x, r_{\mathcal{N}})}{n_{\mathcal{N}} + 1} \tag{4.6}$$

follwed by an increment of $n_{\mathcal{N}}$.

Once a leaf node has been reached, a comparison is made to see if the vector to be added is already present in the leaf node (or is sufficiently close to the vector stored in the leaf node). If it is, the usage count of the leaf is incremented and no more is done. Otherwise, two child nodes are spawned from the old leaf node. One of these has the data from the old leaf copied down and the other has the new data placed in it. The average distortion between the old leaf node and the two new children is then calculated and stored in the old leaf node, followed by the hyperplane normal direction and intersection point. It is worth noting that if the old leaf has a usage count greater than one, this is copied down to the duplicate child node.

In summary, adding a vector to the tree proceeds in the following way.

- If the tree is empty
  - Create new root node containing the supplied data. Nothing more is done.
- If the tree is non-empty
  - At each node, evaluate $2x^T z_{\mathcal{N}}$ to decide which path to follow. Update

the paramaters in each node passed by equation 4.6. Repeat until a leaf node is reached.

- At the leaf node, check if $d(x, z_{\mathcal{N}}) < T$, for some threshold, T. If so, increase the usage count of the node and finish
- If $d(x, z_{\mathcal{N}}) \geq T$, create two nodes subordinate to $\mathcal{N}$. Copy the datum and other parameters from node $\mathcal{N}$ to the right hand node and place the new datum in the left hand node.
- Re-calculate the hyperplane parameters for node $\mathcal{N}$

Since the separating hyperplanes are fixed by their placement between the first two vectors subordinate to a node, it is quite likely that the tree will become unbalanced. To combat this, a method for restructuring subtrees has been designed, based on the Linde-Buzo-Gray vector quantizer design algorithm [LBG80].

4.3.1 The Linde-Buzo-Gray Vector Quantizer Design Algorithm

The LBG algorithm for vector quantizer design [LBG80] is used to design near-optimal block quantizers with a given number of representative vectors. The quantizer is *trained* by a sequence of training data, usually taken to be representative of the population that the quantizer must encode. The algorithm produces an optimum partition of the training set. This is simply extended into an n-dimensional optimum partition of Euclidean n-space by the nearest neighbour rule.

Given N, the number of representative vectors in the final quantizer, a training sequence $\{x_j : j = 0, \ldots, n-1\}, n \geq N$, and an error threshold $\epsilon \geq 0$, the algorithm proceeds as follows

- Initialisation
  - Given $\{x_j\}$, set $\hat{A}_0(1) = \bar{x}$ where $\bar{x}$ is the centroid of the training sequence.
  - Given $\hat{A}_0(M)$, containing M vectors, perturb each vector in the set by a small, fixed vector, ν. So, each $y_i \in \hat{A}_0(M)$ is replaced by the two vectors $\{y_i + \nu, y_i - \nu\} \in \hat{A}_0(2M)$. Set M=2M.
  - Repeat previous step until $M \geq N$. Then, the resultant set $\hat{A}_0(N)$ is the initial set to which the partitioning algorithm is applied.
- Partitioning
  - Given $\hat{A}_m$, find the minimum distortion partition $\mathcal{P}(\hat{A}_m) = \{C_i : i = 1, \ldots, N\}$ of the training set by assigning each vector x_j to a cluster C_i using the rule

$$x_j \in C_i \Leftrightarrow ||x_j - y_i|| \leq ||x_j - y_l|| \forall l. \tag{4.7}$$

  - Calculate the average distortion

$$D_m = \frac{1}{n} \sum_{j=0}^{n-1} \min_{y \in \hat{A}_m} ||x_j, y||. \tag{4.8}$$

  If $\frac{D_{m-1} - D_m}{D_m} \leq \epsilon$ stop with $\hat{A}_m$ as the final set of representative vectors.

– Re-estimate the centroids of the clusters $\{C_i\}$. Set $\hat{A}_{m+1} = \{\bar{C}_i : i = 1, \ldots, N\}$. Set $m = m + 1$. Repeat the partitioning step.

4.3.2 Restructuring an Unbalanced Tree

Before a restructure is performed, it must first be decided where (or if) the tree is actually unbalanced. Since the usage count in any node represents the number of leaf data nodes below it, a comparison of usage counts of sibling nodes is an appropriate measure of how unbalanced the subtree is. Note that the root node usage count is the total number of leaf node data vectors in the subtree.

A recursive traversal down the tree is performed from the root. At any point, n_{Root}, n_{Left}, n_{Right}, the usage counts in the subtree root node and the current left and right child nodes are known and it is simple to calculate p_R and p_L

$$p_L = \log_2\left(\frac{n_{Root}}{n_{Left}}\right) \tag{4.9}$$

$$p_R = \log_2\left(\frac{n_{Root}}{n_{Right}}\right) \tag{4.10}$$

$$\tag{4.11}$$

and so

$$p_R - p_L = \log_2\left(\frac{n_{Left}}{n_{Right}}\right). \tag{4.12}$$

A subtree is deemed unbalanced if $|p_R - p_L| > 0.5$. Once an unbalanced subtree is found, the leaf data are extracted and the subtree is deleted. The subtree root data value is set to the centroid of all the leaf data. Then, the LBG algorithm is run on the leaf data to produce two cluster centroids. These centroids are used as the data for the child nodes of the subtree root node. The separating hyperplane is calculated and its normal vector and intersection point are stored in the subtree root. The application of the LBG algorithm implicitly generates two new subtree root nodes and two sets of leaf data nodes. These are the two sets of data that constitute the clusters generated by the algorithm. The procedure is applied recursively until only two data vectors remain. Then, a simple split is performed and the subtree is fully restructured. The final procedure in the restructure is to work back up the subtree to the subtree root in order to update the usage counts in each node. This is a simple by-product of the recursive nature of the procedure.

4.3.3 Variable rate coding using the TSVQ

The tree structured vector quantizer can be used to perform variable rate coding. The rate can be varied by either constraining the maximum number of bits used to encode a given vector or by a maximum mean square error per vector. Since each node contains both a representative vector and the average distortion between the vector in the current node and its children, a simple comparison will determine if either the mean square error distortion or the bit rate constraint has been reached.

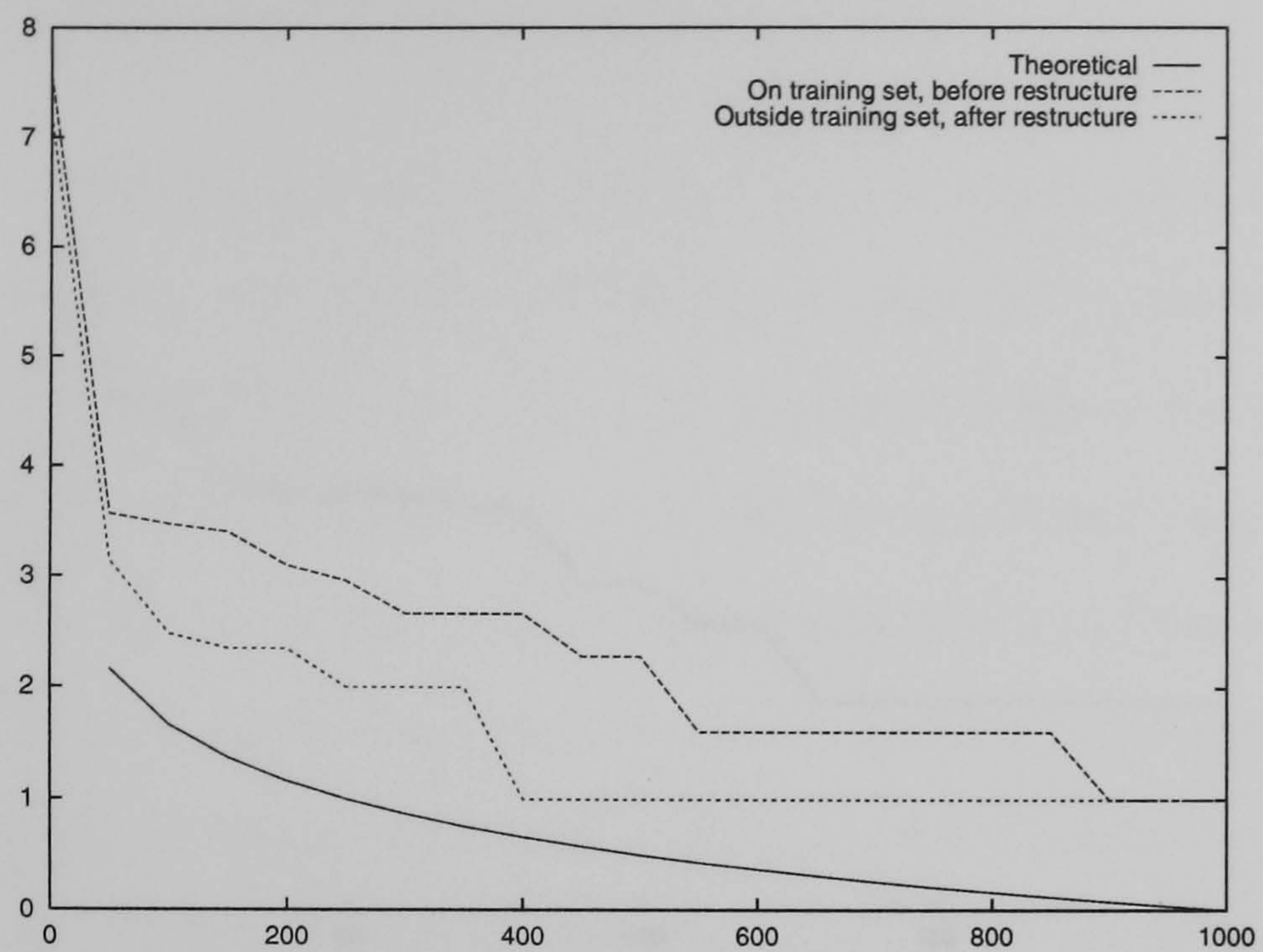


Figure 4.5: One Dimensional Results for the TSVQ, $\sigma^2 = 1000$

4.3.4 Results

The tree-based quantizer was tested on several sets of random data drawn from Gaussian sources of variance σ^2 in one dimension. The results are compared with the theoretical rate $r_d = \frac{1}{2}\log_2\left(\frac{\sigma^2}{d}\right)$ and distortion d in figures 4.5 and 4.6. The 'Outside training set' data is the average over 10 runs, and matches the results when the data is taken from the training set. As can be seen, the Rate-Distortion performance of the quantizer follows the bound well, except at the highest distortions which require rates below 1 bit, which it cannot deliver.

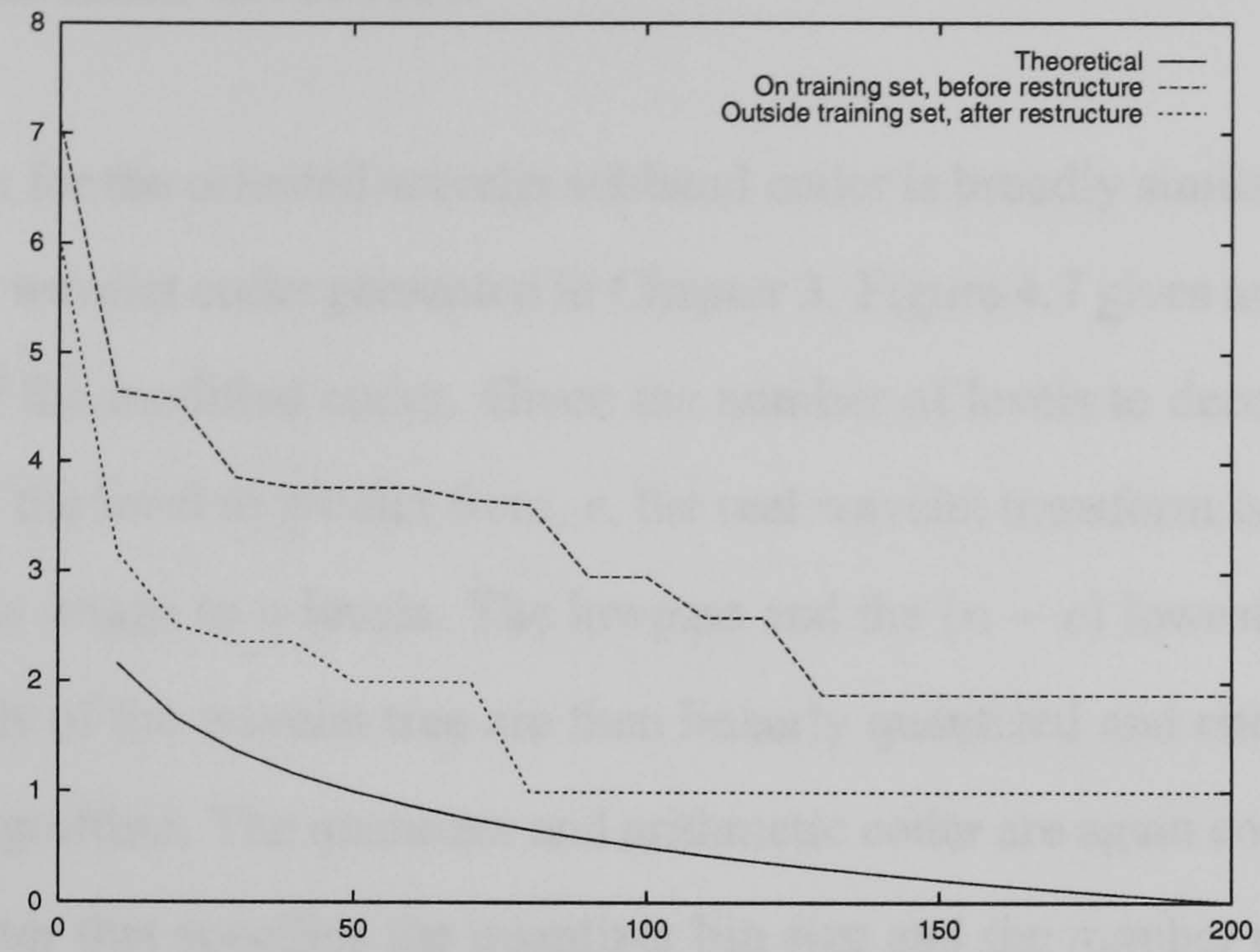


Figure 4.6: One Dimensional Results for the TSVQ, $\sigma^2 = 200$

4.4 An Oriented Wavelet Image Coding Algorithm

The oriented wavelet subband decomposition presented previously refines the orientation selectivity of the wavelet transform without increasing the volume of data produced. The tree structured vector quantizer gives a method of finding nearest-neighbour matches in logarithmic time. The two schemes appear to complement each other well and form the basis of a compression scheme which should be more efficient than that of the previous chapter.

4.4.1 Algorithm Overview

The algorithm for the oriented wavelet subband coder is broadly similar to that of the predictive wavelet coder presented in Chapter 3. Figure 4.7 gives an algorithm description of the modified coder. Given the number of levels to decompose the image, n, and the level to predict from, c, the real wavelet transform is applied to decompose the image to n levels. The lowpass and the $(n - c)$ lowest frequency highpass levels of the wavelet tree are then linearly quantized and encoded as in the original algorithm. The quantizer and arithmetic coder are again controlled by a rate parameter that specifies the quantizer bin size and the number of symbols that the arithmetic coder can encode. The oriented wavelet transform is then applied to the quantized real wavelet data. The encoding process proceeds from one level to the next. It requires two wavelet coefficient levels, one to predict from and one to approximate. Given these levels, the algorithm begins by scanning the six complex-valued wavelet coefficient bands and extracting (possibly overlapping) blocks of data on a p-pixel grid. Any block whose energy is below a threshold is treated as noise and ignored. The coder implements two tree structured vector quantizers, detailed earlier in this chapter. One is used for the horizontal and vertical derived complex bands and one for the diagonal derived complex bands. As suggested in section 4.1, the horizontal and vertical derived subbands are sufficiently related to allow one vector quantization tree to be used for them. The diagonal bands are not related to the others in any simple way and so are treated separately. The extracted blocks are added to the relevant tree structured vector

- Initialization
  - Real wavelet transform to n levels.
  - Linear quantize and entropy code levels $c \leq l \leq n$.
  - Oriented wavelet transform on levels 1 to c.
  - Build the initial HV and D VQ trees from the quantized level c using $m \times m$ blocks.
- For each level, $l = c - 1$ to 1
  - For each non overlapping $m \times m$ block on level l in each of the six oriented bands
    - ∗ Find the nearest matching block in the appropriate VQ tree.
    - ∗ Quantize the scale factor
    - ∗ If quantized scale factor is zero
      - · Clear bit in context and encode a zero in the usable map stream
    - ∗ else
      - · Set bit in context and encode a one in the usable map stream
      - · Send the path down the VQ tree and entropy code the quantized scale factor
  - Add level l to the HV and D VQ trees.

Figure 4.7: Algorithm Description of the Complex Wavelet Coder

quantizer, after being transformed into a canonical orientation (relative to the appropriate tree) and normalized. The canonical orientations are taken, arbitrarily, as the orientation of band 1 for the HV tree and that of band 3 for the D tree. Since the tree structured vector quantizers are populated from data known to both the encoder and decoder, explicit transmission of the codebooks is avoided. The encoding process for a given level is simple. The six complex sub-bands are partitioned into non-overlapping blocks,as in figure 4.8. The shaded bands are not coded and will be reconstructed using the hermitian symmetry property of the transform. The blocks from the same relative position in each of the six bands are encoded in turn. That is, for each band, block number 1 is encoded, followed by block number 2 and so on. The blocks are transformed into the canonical orientation of the appropriate vector quantizer tree and the tree is searched for the best matching vector. The search can be terminated by a maximum path length constraint, a minimum error constraint or by finding a leaf node. The constraints on maximum path length and minimum error are configuration parameters of the encoder. A simple inner product between the source vector and the vector found in the tree yields the scale factor. Associated with each block position is a six bit context value. The context consists of 'code/not code' decisions for each oriented wavelet subband. The 'code/not code' decision is based solely on the quantized scale factor: if the quantized scale factor is not zero, the appropriate bit is set in the context map and a *one* flag is sent to the decoder, indicating that parameters are present for this prediction. If the scale factor quantizes to zero, the appropriate bit in the context map is cleared and a *zero* flag is sent to the decoder. If the block

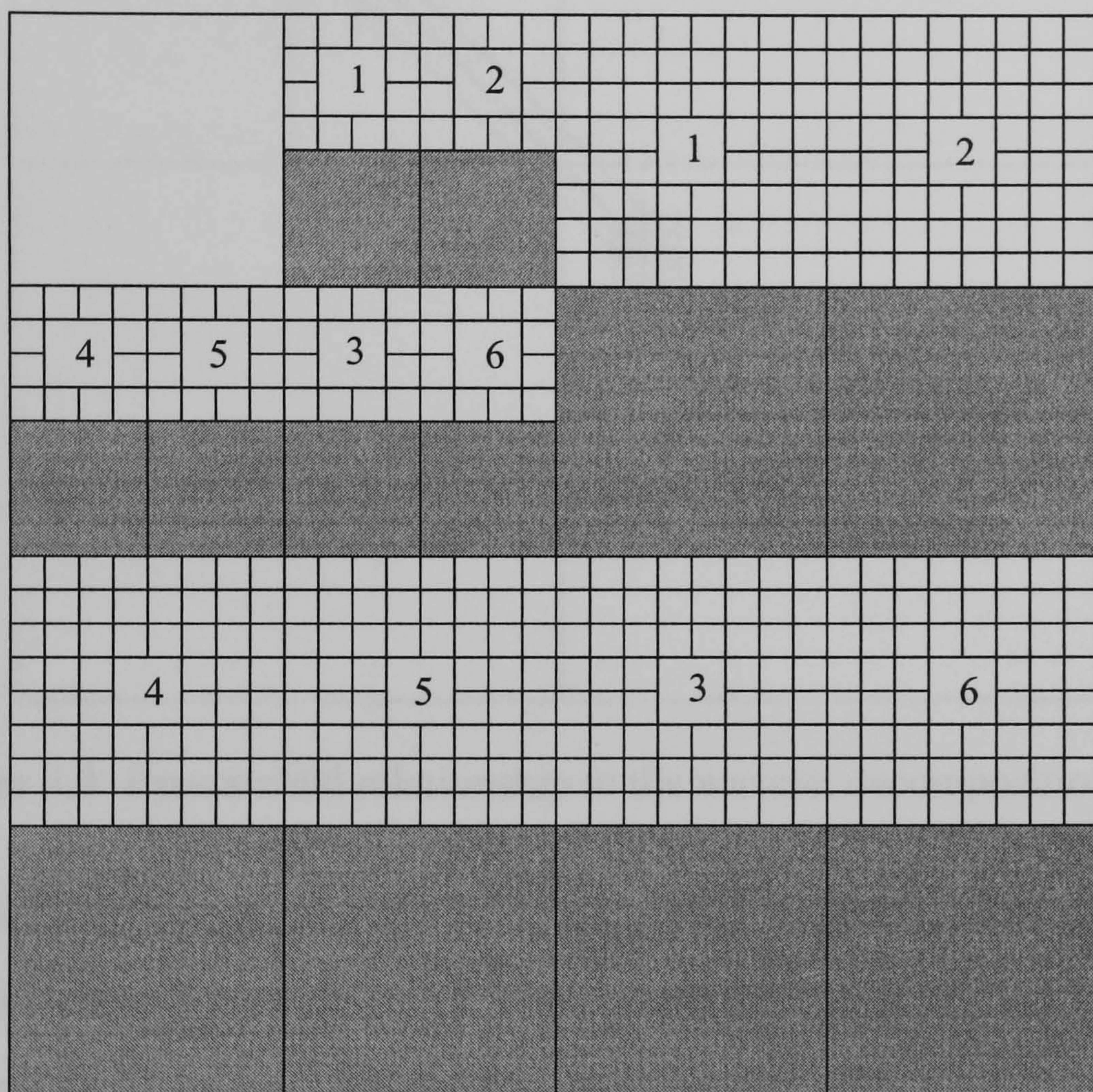


Figure 4.8: Tessellation of the oriented wavelet sub-bands

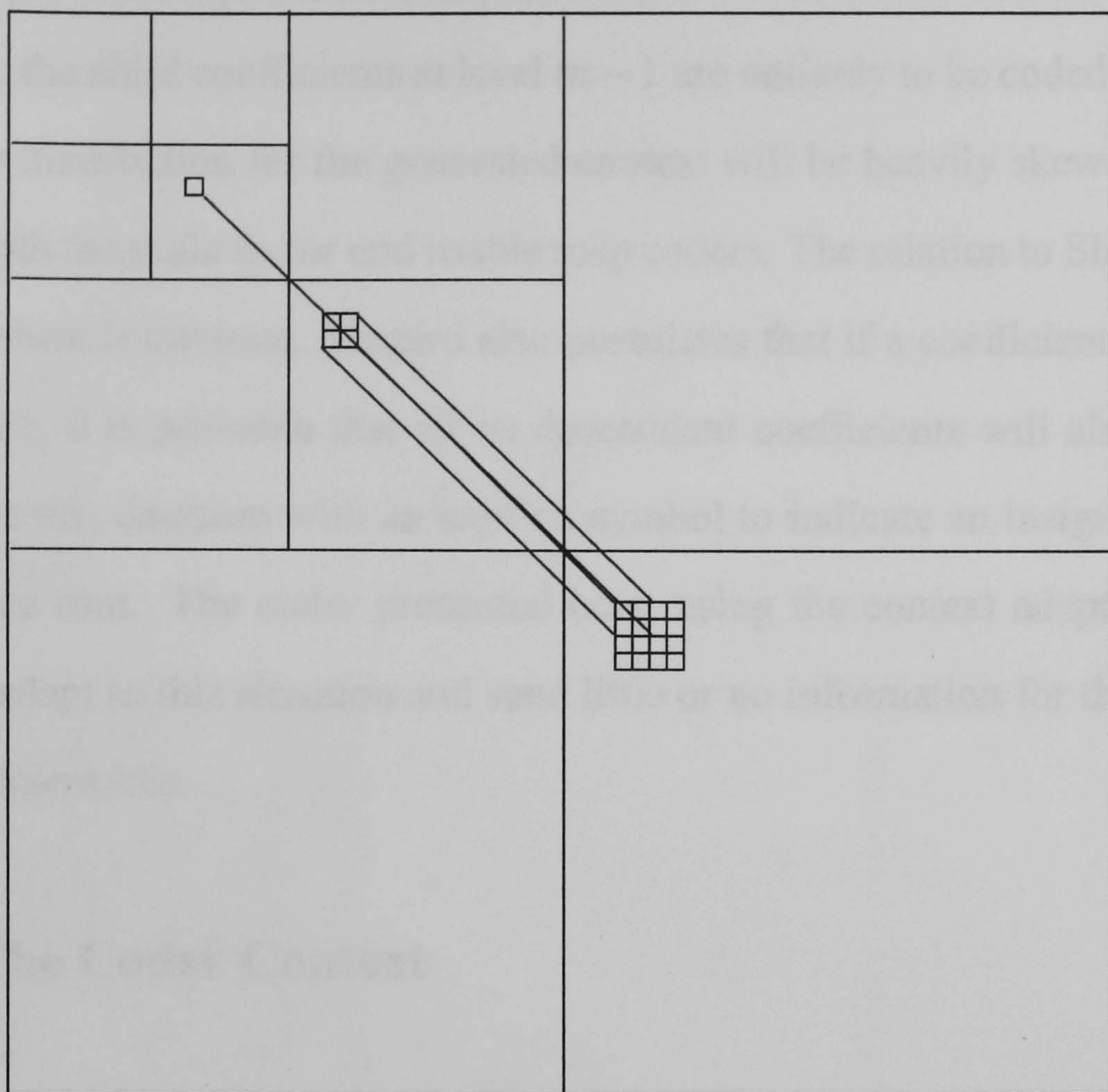


Figure 4.9: Parent/child relationship in the wavelet decomposition tree

is to be coded, the quantized scale factor and path down the VQ tree are coded by a context-weighted arithmetic coder. The context value is taken from the position of the parent block in the wavelet decomposition tree, as in figure 4.9. The context value is used internally by the arithmetic coder to switch probability tables. Since only the probability table referenced by the context value is updated during a symbol encoding operation, each table will adapt differently, depending on how the context is constructed.

This context adaptation from the parent coefficient mimics Shapiro's zerotree coding [Sha93]. For example, if the component coefficients are not coded at a position on level m, the child coefficients at level $m-1$ are unlikely to be coded. Thus, the probability distribution for the generated context will be heavily skewed towards zero, for both the scale factor and usable map coders. The relation to Shapiro's zerotree structure is obvious. Shapiro also postulates that if a coefficient at a given scale is zero, it is probable that all its descendant coefficients will also be zero. He encodes this decision with an explicit symbol to indicate an insignificant coefficient tree root. The coder presented here, using the context adaptation, will implicitly adapt to this situation and send little or no information for the insignificant coefficient tree.

4.4.2 The Coder Context

Visual representations of the coder context are shown in figure 4.10 for both high and low bit rates, resulting from coding the *Lena* image. In the figures, which have been resized for ease of interpretation, a black pixel indicates that the block originating at that point was coded in at least one of the sub-bands, and white indicates that the block was ignored. Figures 4.10(a) and (b) show the context that results from encoding level 2 in the *Lena* image using fine ($c = 16, q = 8, 0.315bpp, 30.26dB$) and coarse ($c = 64, q = 64, 0.074bpp, 27.21dB$) quantizers. Figures 4.10(c) and (d) show similar results for level 1. At the high bit rates, more information is expended coding the higher frequency areas of the image, like the hair and feathers. At lower bit rates, it is obvious that these areas are

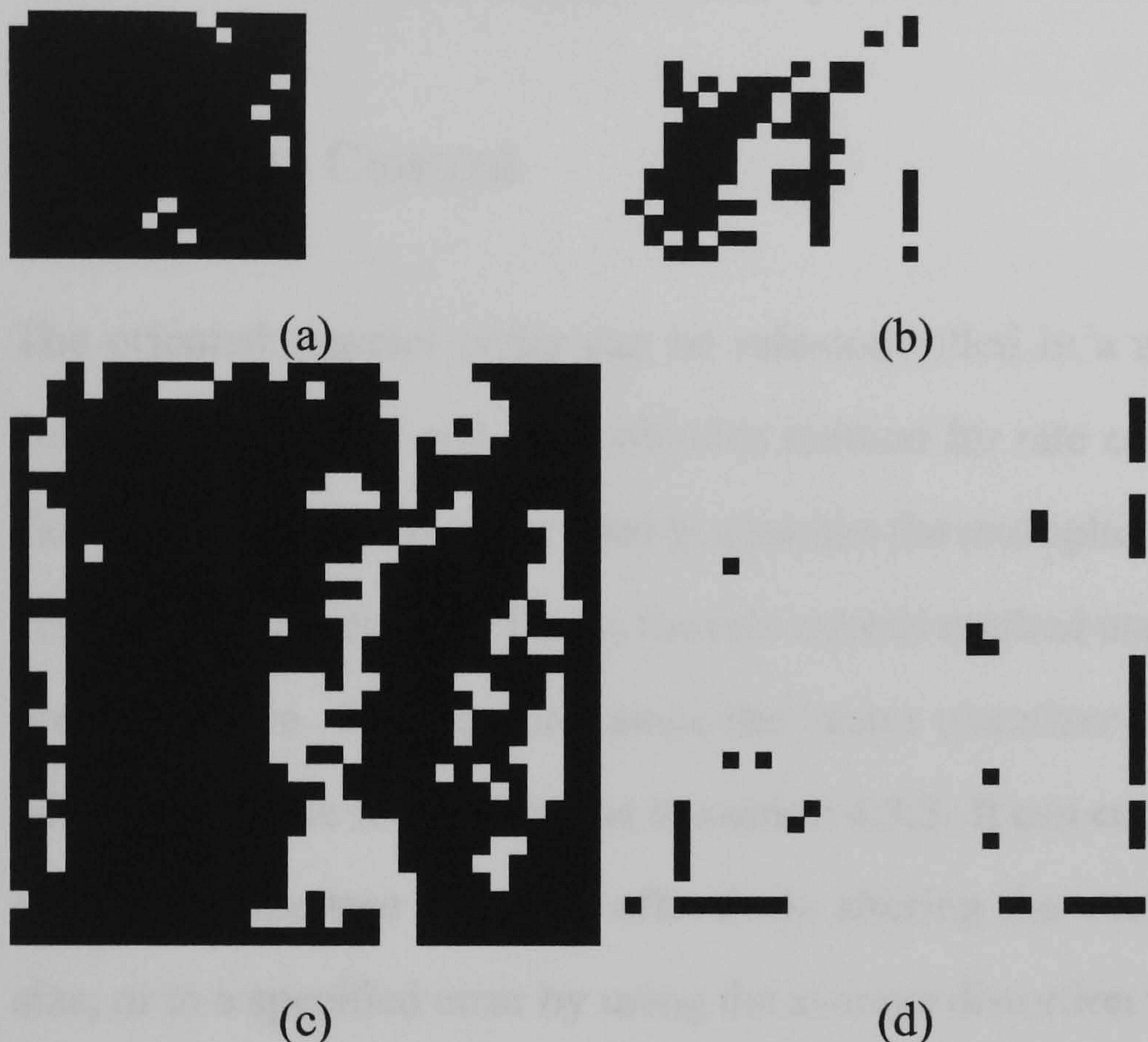


Figure 4.10: Visual representation of the final coder context. (a) Level 2, fine quantizers, (b) Level 2, coarse quantizers, (c) Level 1, fine quantizers, (d) Level 1, coarse quantizers. Black indicates a decision to code the block at that position.

forgone for the more dominant features. Also, the quadtree/zerotree type structure is demonstrated by these diagrams. If a block is not coded on level 2, it is rarely coded on level 1. The adaptive context used by this system appears to emulate Shapiro's zerotree structure reasonably well and also agrees with the intuitive idea of scale invariance of edge features [ZM90].

4.4.3 Rate Control

The oriented wavelet coder can be rate-controlled in a number of independent ways. The simplest and most obvious method for rate control is to alter the linear quantizer bin size that is used to quantize the multiplicative scale factors of the vector quantizer entries. This is the rate control method used in most of the results presented here. Furthermore, since the vector quantizer is tree structured, it can perform variable rate coding, as in section 4.3.3. It can code to a specified bit rate by limiting the tree traversal, effectively altering the vector quantizer codebook size, or to a specified error by using the average distortion value contained in each node. The quantizer used to scalar quantize the lower frequency wavelet coefficients can also be varied as can the dimensionality of the vectors. The combined parameter space for the coder is far too large to explore fully, but a number of tests have been performed to identify useful values.

4.5 Results

The complex subband predictive wavelet coder was applied to a variety of images, using a variety of wavelet filters and coding parameters. The results are presented in this section. The majority of the tests were performed on the *Lena* image with 8-bit greyscale and 512×512 pixels, to investigate the effects of the wavelet filters used on the reconstruction error and bit rate. For comparison and completeness, certain coder configurations were tested on other images. Wall clock times for execution of the encoder and decoder are also given for each of the coder configurations in seconds [1]. [2]

4.5.1 Configuration 1

The encoder and decoder were configured as follows

- Base wavelet level 5
- Start prediction from level 3
- Block size is 4×4
- Low-energy blocks are skipped
- Pre-load phase uses a 2-pixel grid

[1] Based on execution upon a Sun Ultra 1, 143 MHz using no code optimizations. The machine was not fully quiescent while the test were being performed.

[2] Times given do not include the actual loading or saving of the image data.

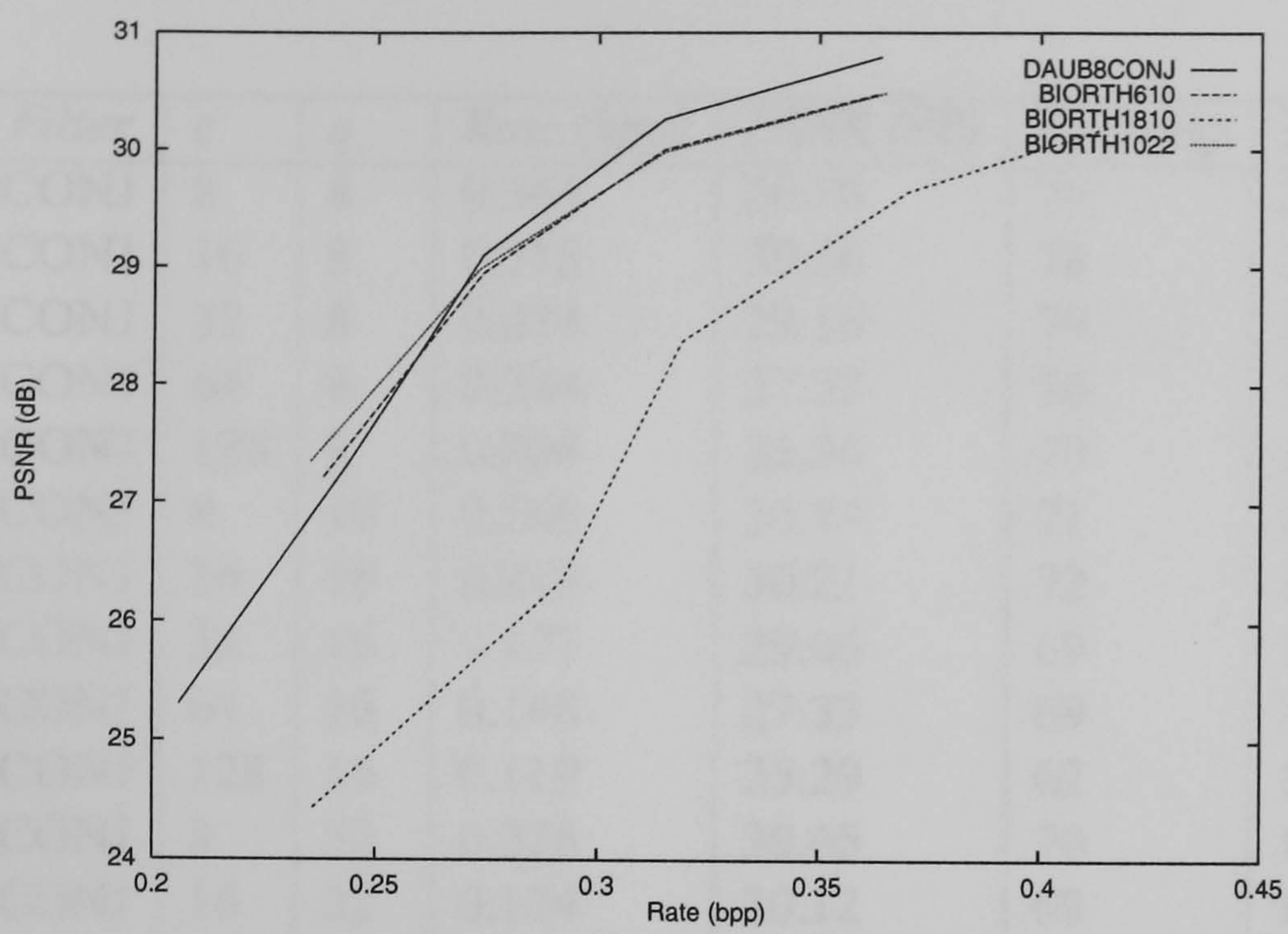


Figure 4.11: Results from coder configuration 1, very fine scale factor quantizer step size ($q = 8$)

- TSVQ is unconstrained

The wavelet filters and quantizer granularity for the low frequency wavelet coefficients, c, and VQ scale factors, q, are varied to alter the rate. The encoding and decoding times, in seconds, are denoted T_e and T_d respectively. The results are given in tables 4.6 to 4.9. Rate distortion curves comparing filters in various configurations are given in figures 4.11 to 4.15. Figure 4.16 shows the effect of different wavelet filters on reconstruction of the *Lena* image at a given approximate bit rate. Figure 4.17 shows the *Lena* image coded at various bit rates using the $DAUB8CONJ$ filter. Similar figures, graphs and reconstructions are shown for *Boats* and *Barbara* in the ensuing figures.

Wavelet Filter	c	q	*Rate (bpp)*	*PSNR (dB)*	T_e *(secs)*	T_d *(secs)*
DAUB8CONJ	8	8	0.364	30.79	79	20
DAUB8CONJ	16	8	0.315	30.26	78	21
DAUB8CONJ	32	8	0.274	29.10	79	19
DAUB8CONJ	64	8	0.244	27.35	76	17
DAUB8CONJ	128	8	0.206	25.30	70	14
DAUB8CONJ	8	16	0.269	30.74	71	18
DAUB8CONJ	16	16	0.219	30.21	72	19
DAUB8CONJ	32	16	0.177	29.06	69	17
DAUB8CONJ	64	16	0.148	27.33	69	17
DAUB8CONJ	128	16	0.119	25.29	62	14
DAUB8CONJ	8	32	0.226	30.65	70	18
DAUB8CONJ	16	32	0.174	30.12	69	18
DAUB8CONJ	32	32	0.133	29.00	69	18
DAUB8CONJ	64	32	0.102	27.29	70	17
DAUB8CONJ	128	32	0.077	25.27	58	13
DAUB8CONJ	8	64	0.199	30.49	65	15
DAUB8CONJ	16	64	0.147	29.98	63	15
DAUB8CONJ	32	64	0.106	28.90	63	15
DAUB8CONJ	64	64	0.074	27.21	62	14
DAUB8CONJ	128	64	0.051	25.22	54	12
DAUB8CONJ	8	256	0.183	30.14	54	10
DAUB8CONJ	16	256	0.134	29.71	54	10
DAUB8CONJ	32	256	0.092	28.70	56	9
DAUB8CONJ	64	256	0.059	27.04	54	10
DAUB8CONJ	128	256	0.037	25.14	47	9
DAUB8CONJ	8	512	0.183	30.13	56	10
DAUB8CONJ	16	512	0.133	29.71	55	9
DAUB8CONJ	32	512	0.092	28.70	54	10
DAUB8CONJ	64	512	0.059	27.04	53	9
DAUB8CONJ	128	512	0.037	25.14	47	8

Table 4.6: Results for Coder Configuration 1, *Lena* image.

Wavelet Filter	*c*	*q*	*Rate (bpp)*	*PSNR (dB)*	T_e *(secs)*	T_d *(secs)*
BIORTH610	8	8	0.363	30.47	74	18
BIORTH610	16	8	0.315	30.00	73	19
BIORTH610	32	8	0.274	28.95	72	17
BIORTH610	64	8	0.238	27.21	66	15
BIORTH610	8	16	0.266	30.41	66	16
BIORTH610	16	16	0.217	29.96	66	17
BIORTH610	32	16	0.179	28.91	65	15
BIORTH610	64	16	0.142	27.19	60	14
BIORTH610	8	32	0.219	30.32	63	17
BIORTH610	16	32	0.171	29.88	64	17
BIORTH610	32	32	0.130	28.85	63	15
BIORTH610	64	32	0.098	27.15	59	14
BIORTH610	8	64	0.191	30.18	59	14
BIORTH610	16	64	0.142	29.73	59	14
BIORTH610	32	64	0.101	28.74	59	13
BIORTH610	64	64	0.069	27.07	54	12
BIORTH610	8	256	0.178	29.91	49	8
BIORTH610	16	256	0.129	29.53	49	8
BIORTH610	32	256	0.087	28.56	49	8
BIORTH610	64	256	0.057	26.96	47	8
BIORTH610	8	512	0.177	29.90	49	9
BIORTH610	16	512	0.129	29.50	49	8
BIORTH610	32	512	0.087	28.54	49	8
BIORTH610	64	512	0.056	26.94	46	8

Table 4.7: Results for Coder Configuration 1, *Lena* image.

Wavelet Filter	c	q	*Rate (bpp)*	*PSNR (dB)*	T_e *(secs)*	T_d *(secs)*
BIORTH1022	8	8	0.365	30.48	83	22
BIORTH1022	16	8	0.315	29.98	84	22
BIORTH1022	32	8	0.273	28.98	84	21
BIORTH1022	64	8	0.235	27.34	82	21
BIORTH1022	8	16	0.267	30.44	77	21
BIORTH1022	16	16	0.216	29.94	77	21
BIORTH1022	32	16	0.175	28.94	76	20
BIORTH1022	64	16	0.137	27.32	75	20
BIORTH1022	8	32	0.227	30.37	73	20
BIORTH1022	16	32	0.175	29.87	73	20
BIORTH1022	32	32	0.132	28.89	75	19
BIORTH1022	64	32	0.099	27.28	72	19
BIORTH1022	8	64	0.206	30.24	70	18
BIORTH1022	16	64	0.153	29.75	72	17
BIORTH1022	32	64	0.112	28.79	70	17
BIORTH1022	64	64	0.077	27.22	68	17
BIORTH1022	8	256	0.193	29.96	59	12
BIORTH1022	16	256	0.142	29.57	60	12
BIORTH1022	32	256	0.099	28.61	59	12
BIORTH1022	64	256	0.065	27.10	60	11
BIORTH1022	8	512	0.193	29.97	59	12
BIORTH1022	16	512	0.142	29.57	60	12
BIORTH1022	32	512	0.099	28.61	59	12
BIORTH1022	64	512	0.064	27.10	59	12

Table 4.8: Results for Coder Configuration 1, *Lena* image.

Wavelet Filter	*c*	*q*	*Rate (bpp)*	*PSNR (dB)*	T_e *(secs)*	T_d *(secs)*
BIORTH1810	8	8	0.415	30.19	109	26
BIORTH1810	16	8	0.370	29.64	108	26
BIORTH1810	32	8	0.319	28.37	106	26
BIORTH1810	64	8	0.292	26.35	106	26
BIORTH1810	128	8	0.236	24.43	92	20
BIORTH1810	8	16	0.306	30.16	97	24
BIORTH1810	16	16	0.257	29.61	98	24
BIORTH1810	32	16	0.211	28.34	97	24
BIORTH1810	64	16	0.184	26.34	98	24
BIORTH1810	128	16	0.137	24.42	87	19
BIORTH1810	8	32	0.236	30.10	93	23
BIORTH1810	16	32	0.187	29.56	95	23
BIORTH1810	32	32	0.142	28.31	93	23
BIORTH1810	64	32	0.113	26.31	93	23
BIORTH1810	128	32	0.079	24.41	83	18
BIORTH1810	8	64	0.194	30.00	93	25
BIORTH1810	16	64	0.145	29.46	94	25
BIORTH1810	32	64	0.103	28.25	92	24
BIORTH1810	64	64	0.073	26.27	92	24
BIORTH1810	128	64	0.047	24.39	83	20
BIORTH1810	8	256	0.168	29.87	77	15
BIORTH1810	16	256	0.119	29.31	78	15
BIORTH1810	32	256	0.077	28.16	78	14
BIORTH1810	64	256	0.046	26.20	79	15
BIORTH1810	128	256	0.026	24.36	70	14
BIORTH1810	8	512	0.167	29.84	76	14
BIORTH1810	16	512	0.117	29.32	76	14
BIORTH1810	32	512	0.077	28.13	76	14
BIORTH1810	64	512	0.045	26.17	77	15
BIORTH1810	128	512	0.026	24.36	71	14

Table 4.9: Results for Coder Configuration 1, *Lena* image.

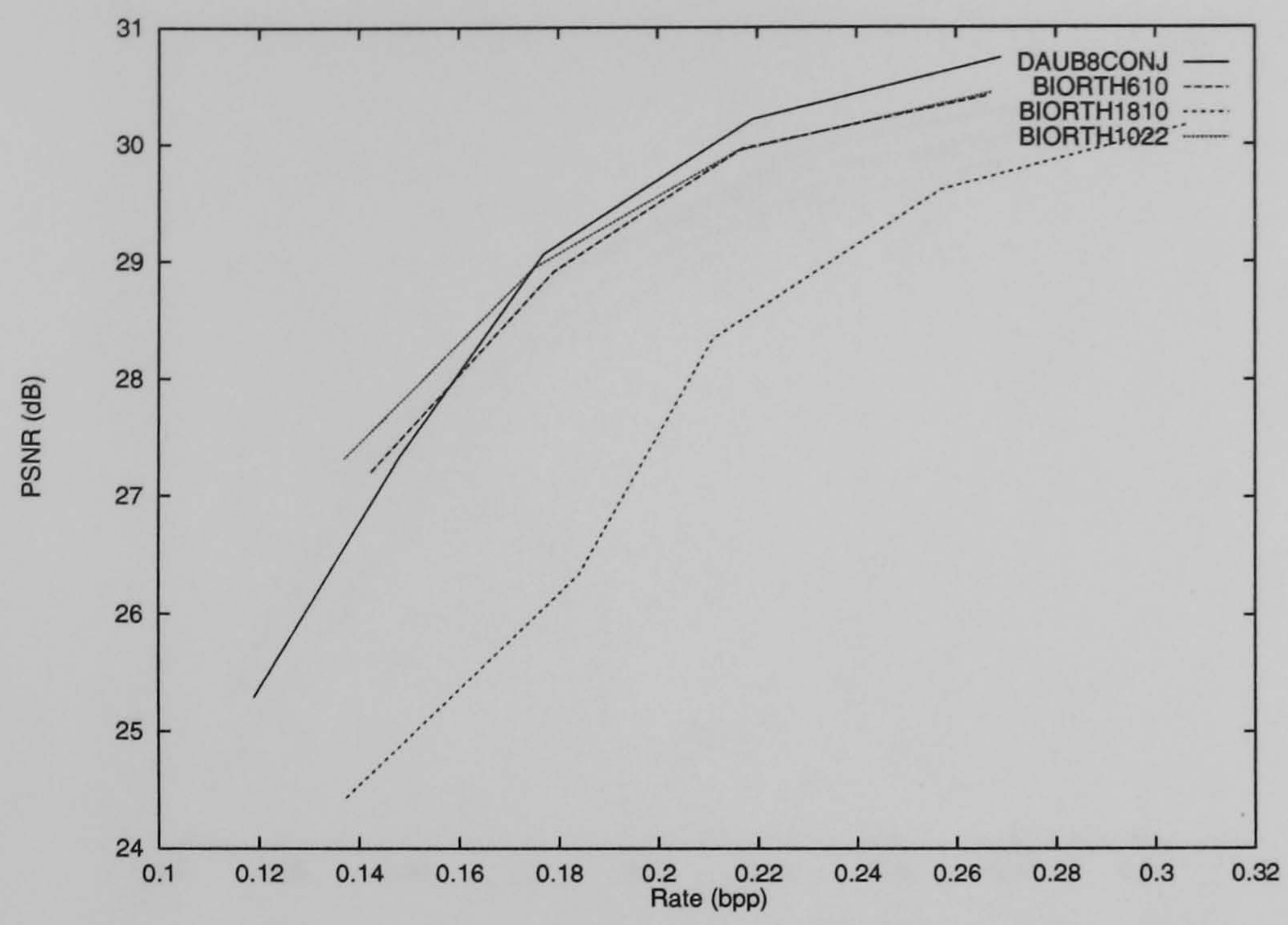


Figure 4.12: Results from coder configuration 1, fine scale factor quantizer step size ($q = 16$)

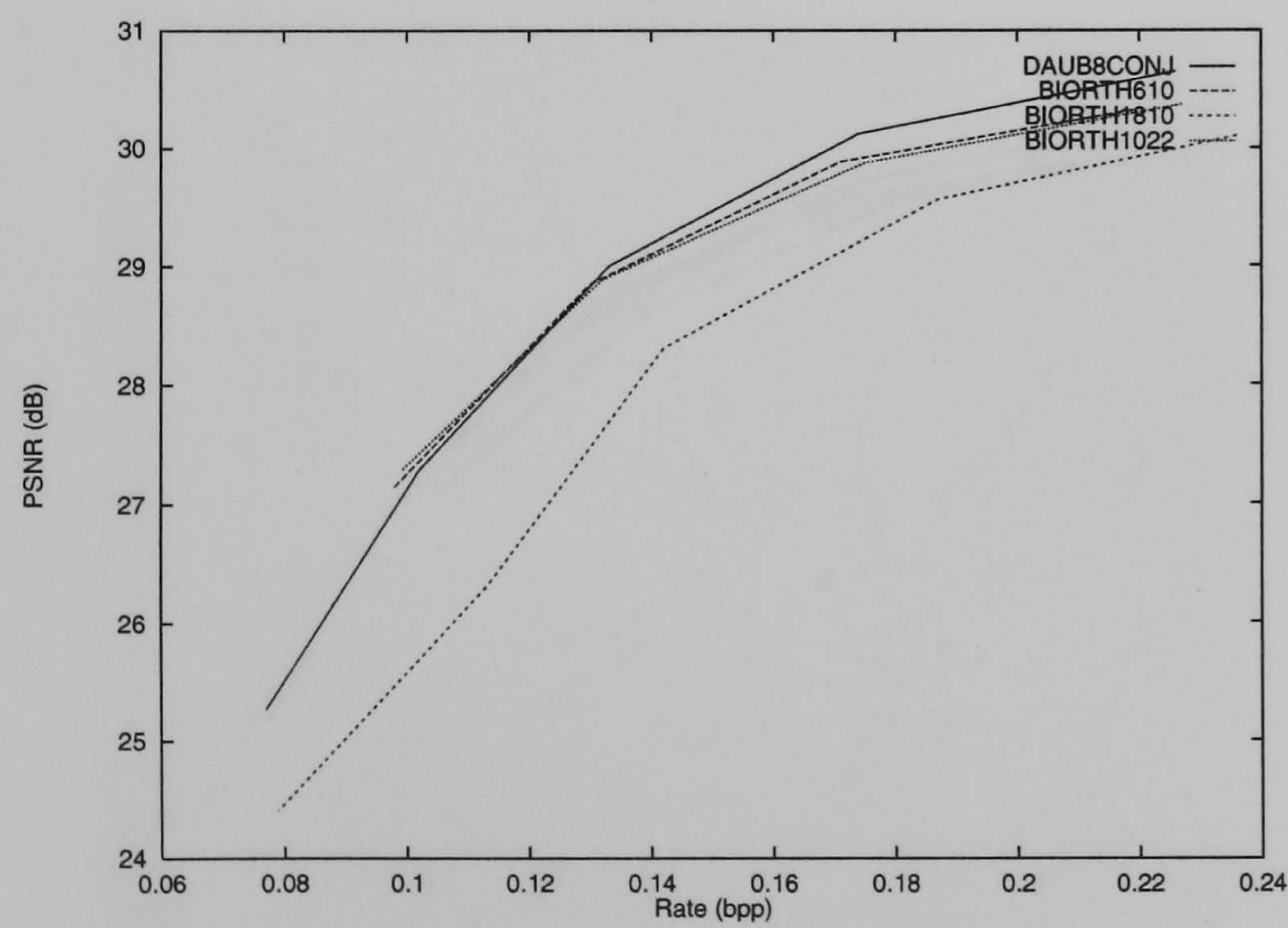


Figure 4.13: Results from coder configuration 1, medium scale factor quantizer step size ($q = 32$)

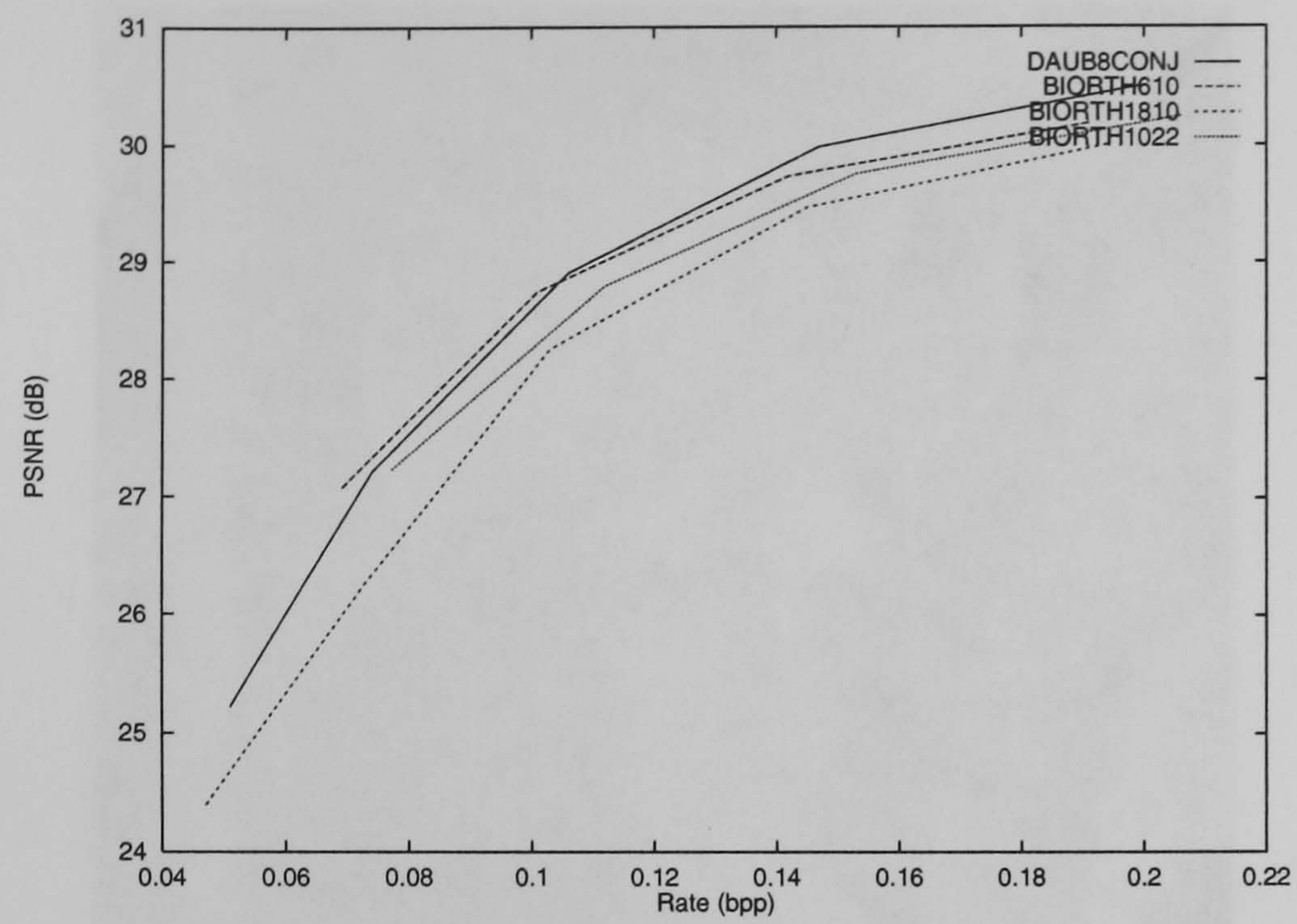


Figure 4.14: Results from coder configuration 1, coarse scale factor quantizer step size ($q = 64$)

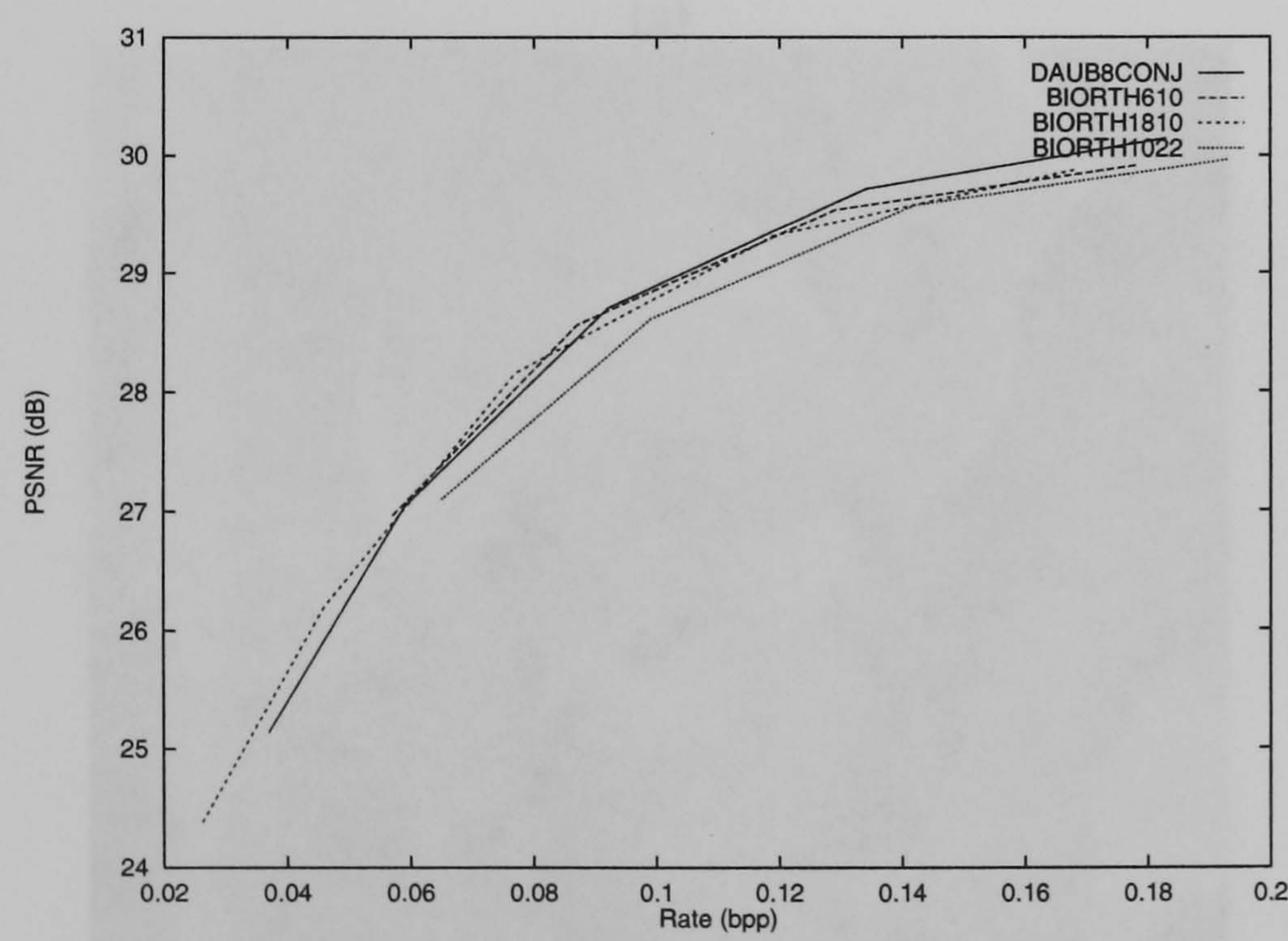


Figure 4.15: Results from coder configuration 1, very coarse scale factor quantizer step size ($q = 256$)

(a)

(b)

Figure 4.16: Reconstructions from coder configuration 1 *Lena* image showing effect of filter on quality; medium scale factor quantizer step size ($c = 32, q = 64$), approx $0.10bpp$ (a) DAUB8CONJ, $28.90dB, 0.106bpp$, (b) BIORTH610, $28.74dB, 0.101bpp$

(c)

(d)

Figure 4.16: Reconstructions from coder configuration 1 *Lena* image showing effect of filter on quality; medium scale factor quantizer step size ($c = 32, q = 64$), approx $0.10bpp$ (c) BIORTH1810, $28.25dB, 0.103bpp$, (d) BIORTH1022, $28.79dB, 0.112bpp$

(a)

(b)

Figure 4.17: Reconstructions from coder configuration 1 *Lena* image, $DAUB8CONJ$ filter.(a) Medium quantizer step size ($c = 16, q = 64$), $29.98dB$,$0.147bpp$ (b) Coarse quantizer step size ($c = 32, q = 256$),$28.70dB$, $0.092bpp$

(c)

(d)

Figure 4.17: Reconstructions from coder configuration 1 *Lena* image, $DAUB8CONJ$ filter. (c) Very coarse wavelet quantizer step size, coarse scale factor quantizer step size $(c = 64, q = 64)$, $27.21dB, 0.074bpp$ (d) Very coarse wavelet quantizer and scale factor quantizer step size $(c = 64, q = 512)$, $27.04dB, 0.059bpp$

Wavelet Filter	*c*	*q*	*Rate (bpp)*	*PSNR (dB)*	T_e *(secs)*	T_d *(secs)*
DAUB8CONJ	8	8	0.399	27.92	85	20
DAUB8CONJ	16	8	0.341	27.62	80	19
DAUB8CONJ	32	8	0.293	26.79	79	20
DAUB8CONJ	64	8	0.259	25.28	79	19
DAUB8CONJ	8	16	0.327	27.89	76	19
DAUB8CONJ	16	16	0.274	27.59	75	18
DAUB8CONJ	32	16	0.222	26.77	78	18
DAUB8CONJ	64	16	0.188	25.26	73	21
DAUB8CONJ	8	32	0.269	27.82	83	22
DAUB8CONJ	16	32	0.215	27.53	74	18
DAUB8CONJ	32	32	0.166	26.72	71	19
DAUB8CONJ	64	32	0.129	25.22	84	19
DAUB8CONJ	8	64	0.230	27.68	78	18
DAUB8CONJ	16	64	0.175	27.39	71	18
DAUB8CONJ	32	64	0.127	26.60	67	17
DAUB8CONJ	64	64	0.090	25.15	82	17
DAUB8CONJ	8	256	0.201	27.33	65	9
DAUB8CONJ	16	256	0.147	27.05	55	10
DAUB8CONJ	32	256	0.100	26.35	54	9
DAUB8CONJ	64	256	0.063	24.98	53	9
DAUB8CONJ	8	512	0.200	27.31	56	10
DAUB8CONJ	16	512	0.147	27.04	83	14
DAUB8CONJ	32	512	0.099	26.34	54	10
DAUB8CONJ	64	512	0.062	24.97	52	9

Table 4.10: Results for Coder Configuration 1, *Boats* image

Wavelet Filter	*c*	*q*	*Rate (bpp)*	*PSNR (dB)*	T_e *(secs)*	T_d *(secs)*
DAUB8CONJ	8	8	0.411	24.03	77	18
DAUB8CONJ	16	8	0.354	23.90	77	19
DAUB8CONJ	32	8	0.311	23.57	80	19
DAUB8CONJ	64	8	0.279	22.97	76	18
DAUB8CONJ	128	8	0.253	22.10	68	14
DAUB8CONJ	8	16	0.339	24.01	73	20
DAUB8CONJ	16	16	0.286	23.89	77	19
DAUB8CONJ	32	16	0.245	23.56	74	18
DAUB8CONJ	64	16	0.211	22.97	73	18
DAUB8CONJ	128	16	0.183	22.09	65	15
DAUB8CONJ	8	32	0.286	23.99	70	18
DAUB8CONJ	16	32	0.230	23.87	71	16
DAUB8CONJ	32	32	0.189	23.54	69	17
DAUB8CONJ	64	32	0.156	22.94	69	19
DAUB8CONJ	128	32	0.128	22.07	63	13
DAUB8CONJ	8	64	0.238	23.91	68	16
DAUB8CONJ	16	64	0.183	23.80	66	16
DAUB8CONJ	32	64	0.141	23.46	67	17
DAUB8CONJ	64	64	0.109	22.88	66	16
DAUB8CONJ	128	64	0.081	22.02	63	14
DAUB8CONJ	8	256	0.197	23.60	52	9
DAUB8CONJ	16	256	0.142	23.49	52	9
DAUB8CONJ	32	256	0.098	23.18	55	10
DAUB8CONJ	64	256	0.063	22.62	52	9
DAUB8CONJ	128	256	0.040	21.84	49	9
DAUB8CONJ	8	512	0.194	23.56	52	9
DAUB8CONJ	16	512	0.140	23.45	51	9
DAUB8CONJ	32	512	0.096	23.14	51	9
DAUB8CONJ	64	512	0.060	22.56	50	9
DAUB8CONJ	128	512	0.038	21.82	50	9

Table 4.11: Results for Coder Configuration 1, *Barbara* image

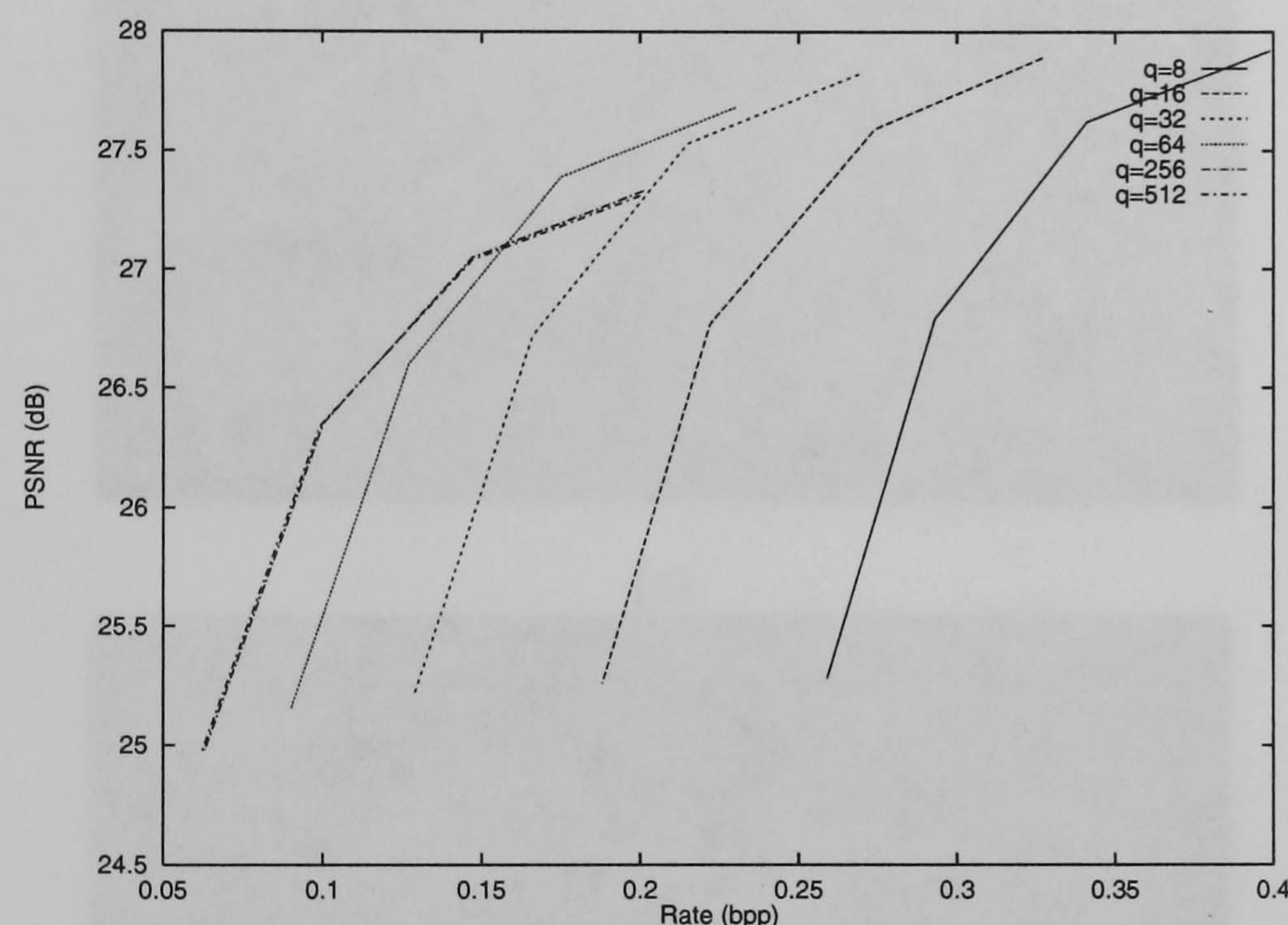
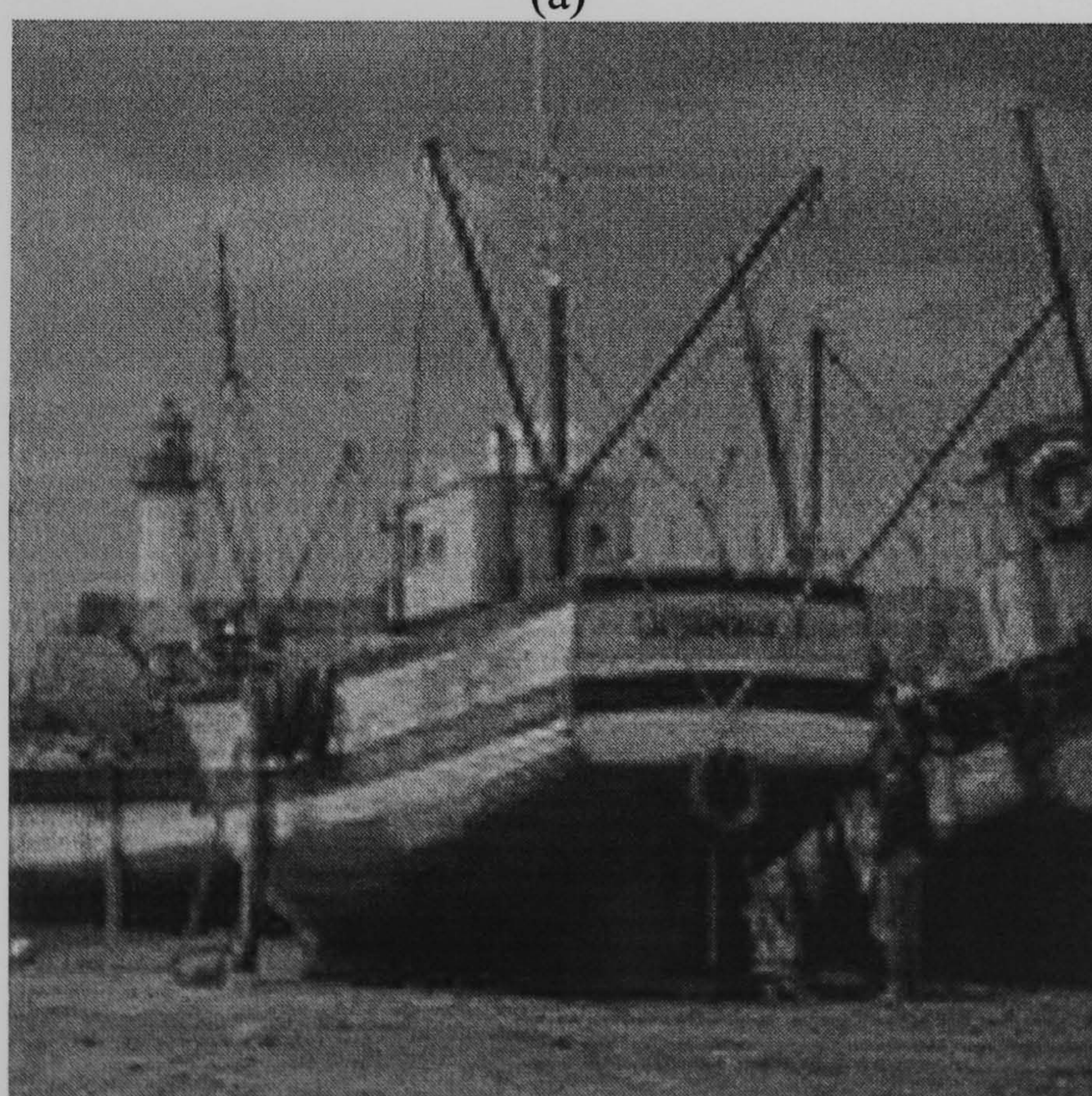


Figure 4.18: Results from coder configuration 1, *Boats* image.

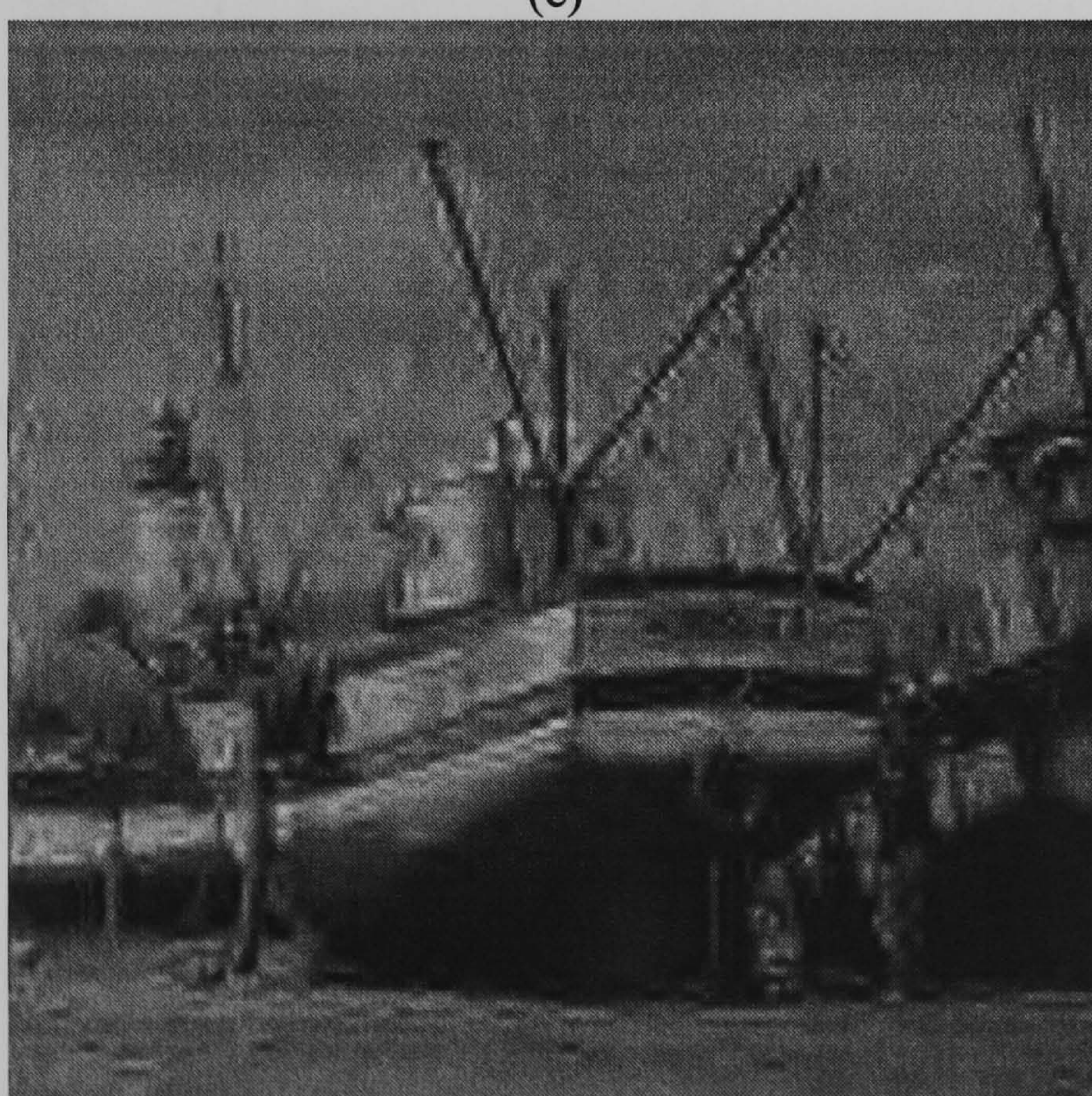
(a)



(b)

Figure 4.19: Reconstructions from coder configuration 1, *Boats* image, DAUB8CONJ.(a) Original (b) Medium wavelet quantizer step size, coarse scale factor quantizer step size ($c = 16, q = 64$), $27.39dB$, $0.175bpp$

(c)



(d)

Figure 4.19: Reconstructions from coder configuration 1, *Boats* image, DAUB8CONJ. (c) Coarse wavelet quantizer step size, coarse scale factor quantizer step size ($c = 32, q = 64$), $26.60dB, 0.127bpp$, (d) Very coarse scale factor and wavelet quantizer step size ($c = 64, q = 256$), $24.98dB, 0.063bpp$

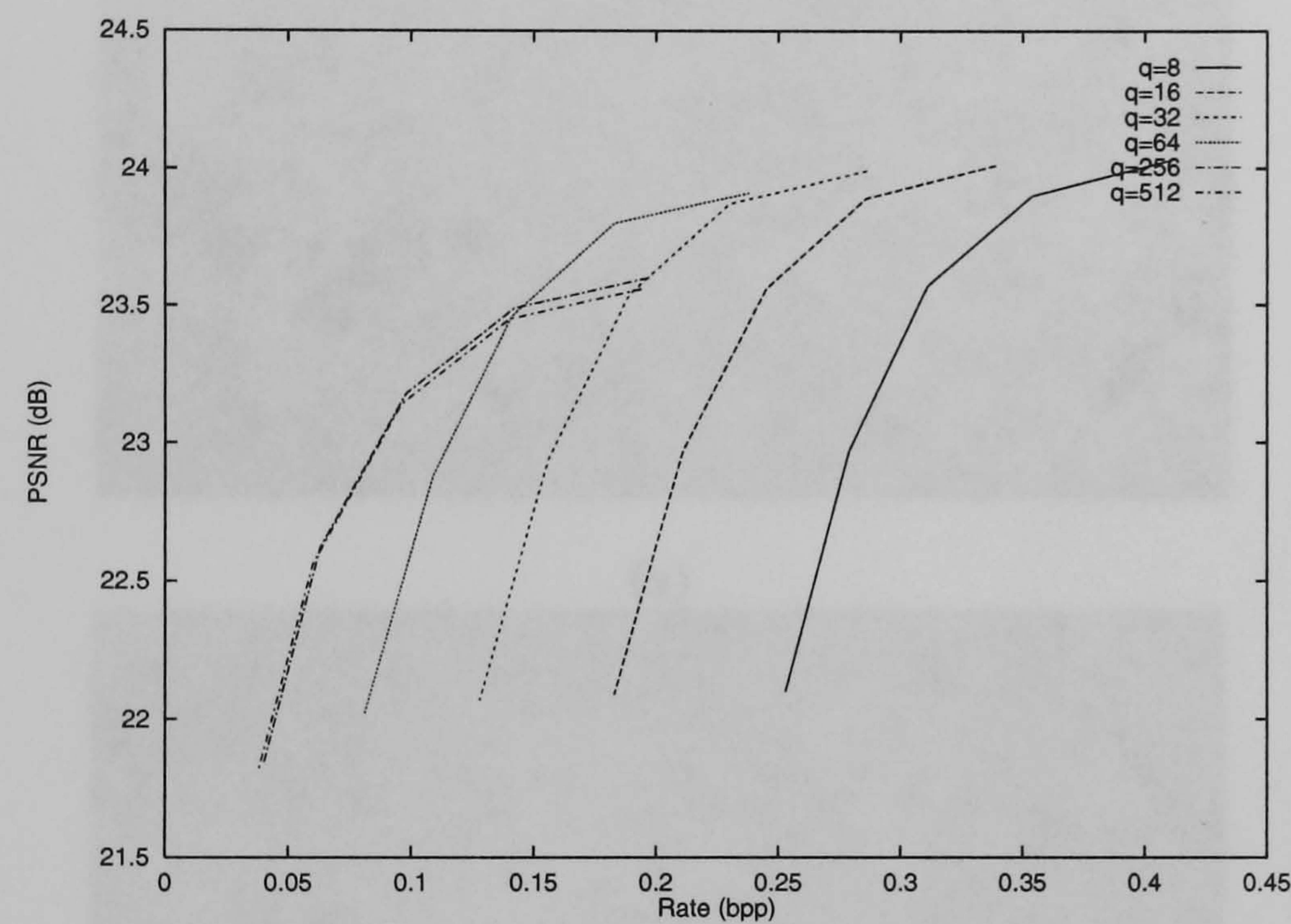
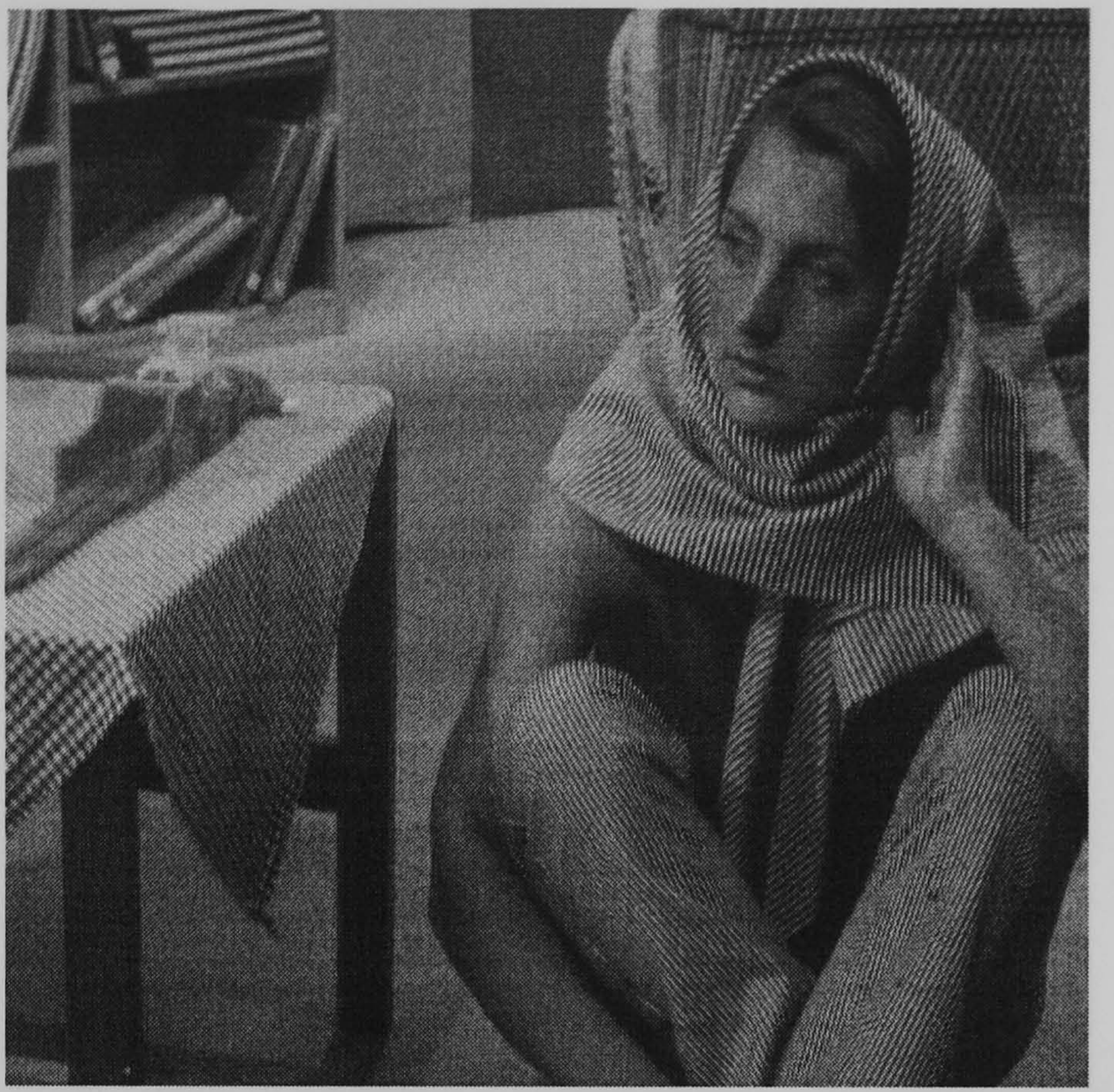


Figure 4.20: Results from coder configuration 1, *Barbara* image.



(a)

(b)

Figure 4.21: Reconstructions from coder configuration 1, *Barbara* image, DAUB8CONJ. (a) Original (b) Medium wavelet quantizer step size coarse scale factor quantizer step size ($c = 32, q = 64$), $23.46dB, 0.141bpp$

(c)

(d)

Figure 4.21: Reconstructions from coder configuration 1, *Barbara* image, DAUB8CONJ. (c) Coarse wavelet quantizer step size, coarse scale factor quantizer step size ($c = 64, q = 64$), $22.88dB, 0.109bpp$, (d) Very coarse wavelet quantizer and scale factor quantizer step size ($c = 128, q = 256$), $21.84dB, 0.040bpp$

Wavelet Filter	*c*	*q*	*Rate (bpp)*	*PSNR (dB)*	T_e *(secs)*	T_d *(secs)*
DAUB8CONJ	32	8	0.136	28.77	111	44
DAUB8CONJ	64	8	0.107	27.11	105	43
DAUB8CONJ	128	8	0.080	25.17	74	28
DAUB8CONJ	32	16	0.114	28.77	94	42
DAUB8CONJ	64	16	0.082	27.11	113	47
DAUB8CONJ	128	16	0.059	25.17	74	30
DAUB8CONJ	32	32	0.102	28.76	94	43
DAUB8CONJ	64	32	0.070	27.10	102	43
DAUB8CONJ	128	32	0.048	25.16	68	27
DAUB8CONJ	32	64	0.095	28.74	80	31
DAUB8CONJ	64	64	0.062	27.08	92	35
DAUB8CONJ	128	64	0.040	25.15	69	21
DAUB8CONJ	32	256	0.092	28.70	54	13
DAUB8CONJ	64	256	0.059	27.04	61	14
DAUB8CONJ	128	256	0.037	25.14	55	14
DAUB8CONJ	32	512	0.092	28.70	57	12
DAUB8CONJ	64	512	0.059	27.04	58	12
DAUB8CONJ	128	512	0.037	25.14	56	12

Table 4.12: Results for Coder Configuration 2, *Lena* image

4.5.2 Configuration 2

The encoder and decoder were configured as previously, except that the block size was set to 8×8. For brevity, only the results from the $DAUB8CONJ$ filter will be presented at the lower bit rates.

4.5.3 Configuration 3

In this section the effect of the variable depth tree structured quantizer will be investigated. The *Lena* and *Barbara* images were coded using a medium/fine

Wavelet Filter	c	q	*Rate (bpp)*	*PSNR (dB)*	T_e *(secs)*	T_d *(secs)*
DAUB8CONJ	32	8	0.148	23.22	91	32
DAUB8CONJ	64	8	0.113	22.64	100	26
DAUB8CONJ	128	8	0.091	21.90	69	22
DAUB8CONJ	32	16	0.131	23.21	81	29
DAUB8CONJ	64	16	0.097	22.64	78	26
DAUB8CONJ	128	16	0.075	21.90	68	22
DAUB8CONJ	32	32	0.118	23.21	88	33
DAUB8CONJ	64	32	0.082	22.64	79	29
DAUB8CONJ	128	32	0.061	21.89	65	23
DAUB8CONJ	32	64	0.105	23.20	72	27
DAUB8CONJ	64	64	0.070	22.62	74	27
DAUB8CONJ	128	64	0.049	21.88	64	23
DAUB8CONJ	32	256	0.096	23.14	62	13
DAUB8CONJ	64	256	0.060	22.57	51	11
DAUB8CONJ	128	256	0.038	21.83	49	12
DAUB8CONJ	32	512	0.095	23.14	46	10
DAUB8CONJ	64	512	0.060	22.57	50	11
DAUB8CONJ	128	512	0.037	21.82	44	10

Table 4.13: Results for Coder Configuration 2, *Barbara* image

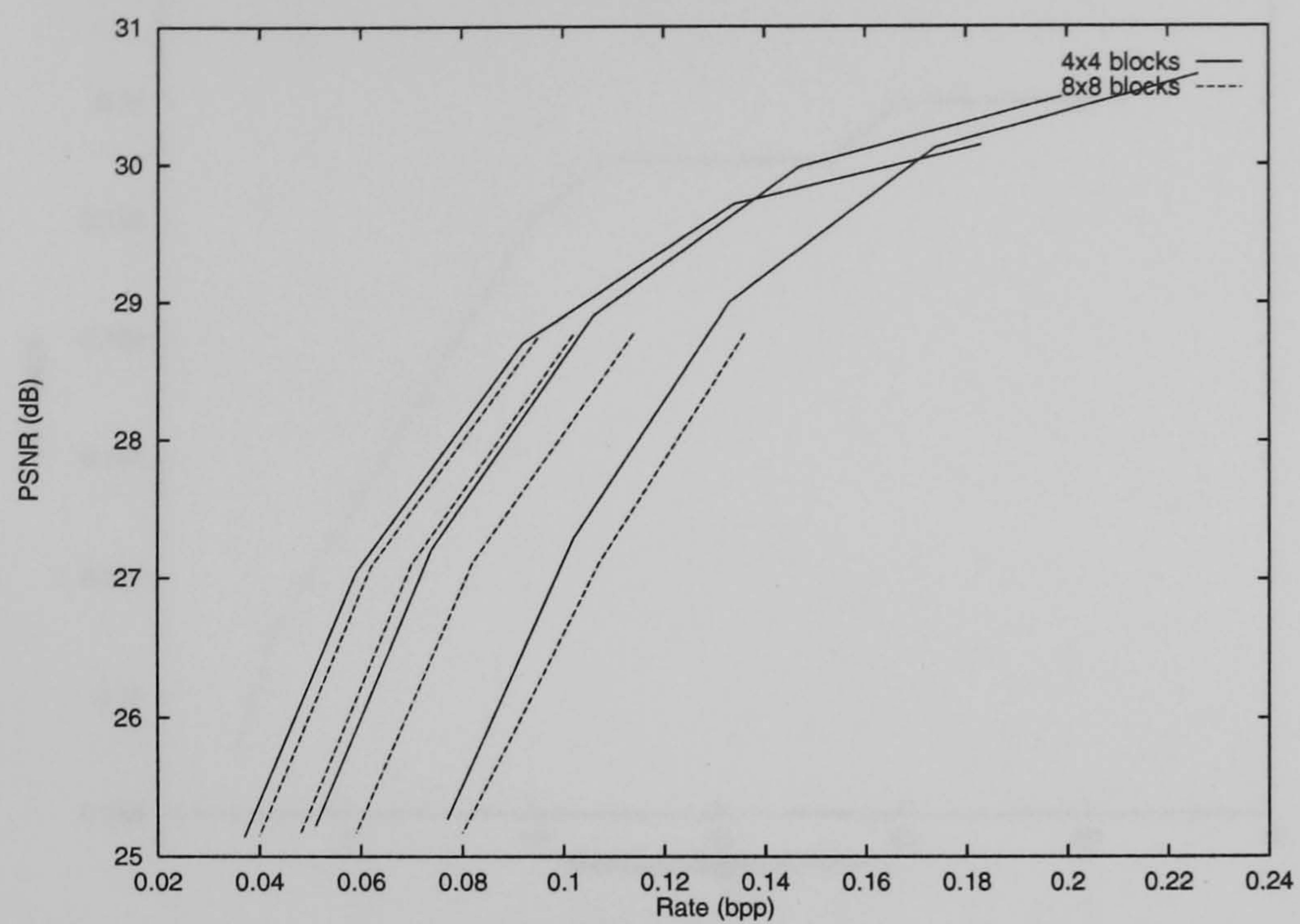


Figure 4.22: Effect of block size on coder performance, *Lena* image.

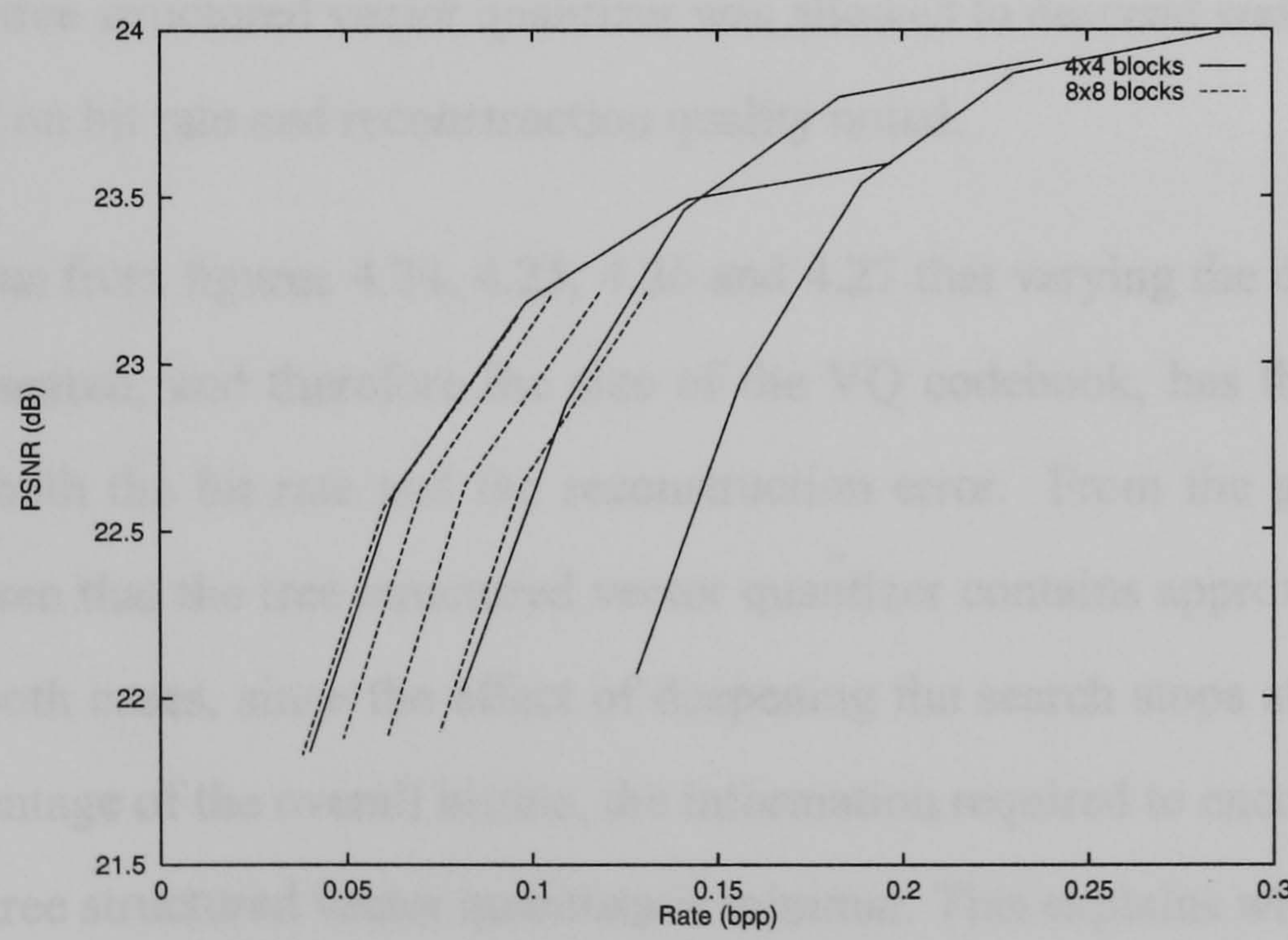


Figure 4.23: Effect of block size on coder performance, *Barbara* image.

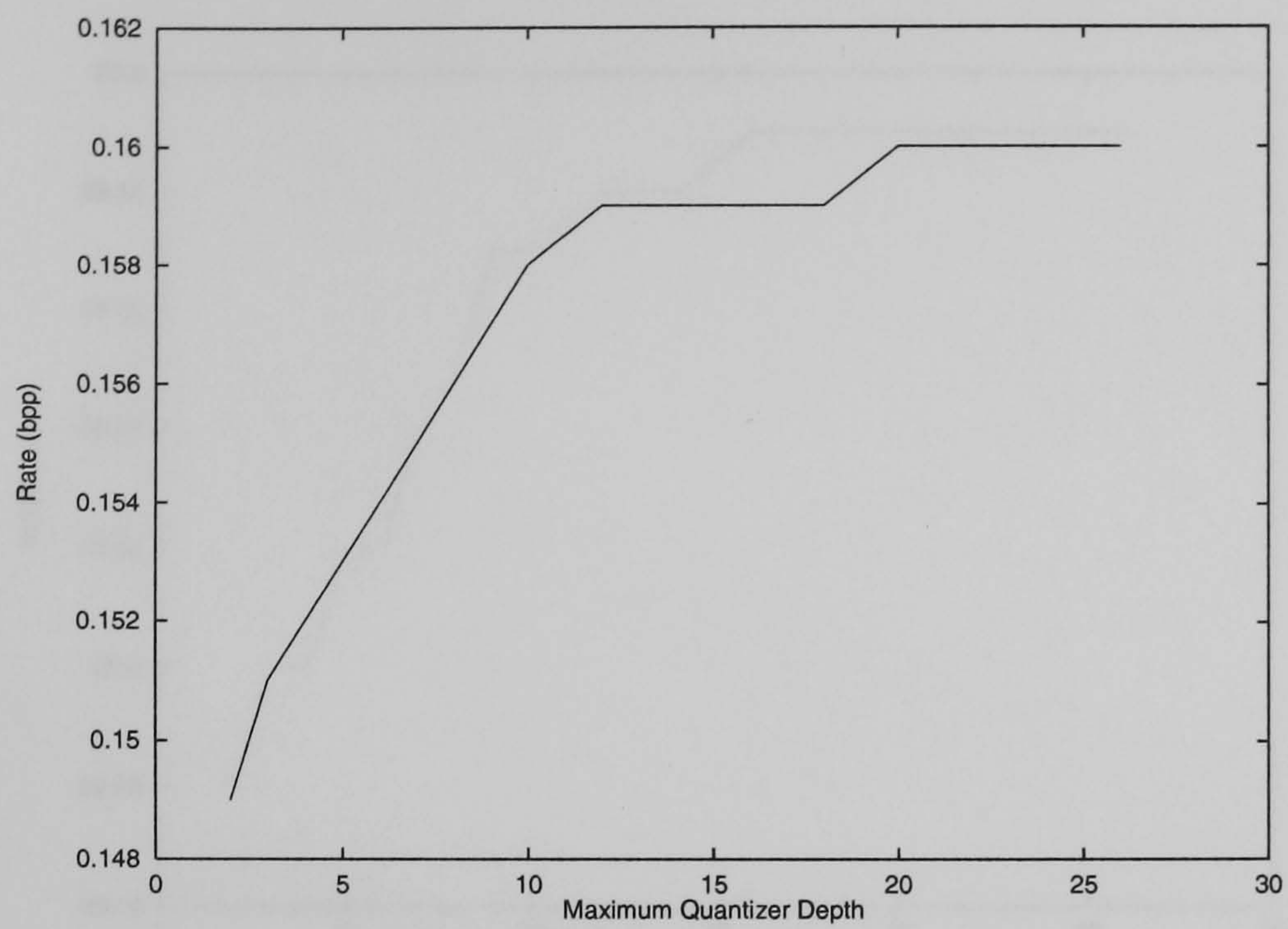


Figure 4.24: Effect of quantizer search depth on bit rate, *Lena* image.

quantizer step size ($c = 16, q = 16$) using the DAUB8CONJ filter. The depth to which the tree structured vector quantizer was allowed to descend was varied and the impact on bit rate and reconstruction quality noted.

It is obvious from figures 4.24, 4.25, 4.26 and 4.27 that varying the depth of the quantizer search, and therefore the size of the VQ codebook, has the expected effect on both the bit rate and the reconstruction error. From the graphs, it is simple to see that the tree structured vector quantizer contains approximately 21 levels in both cases, since the effect of deepening the search stops at this point. As a percentage of the overall bitrate, the information required to encode the path down the tree structured vector quantizer is minimal. This explains why the range of the ordinal axis is disproportionate to the range of search depth.

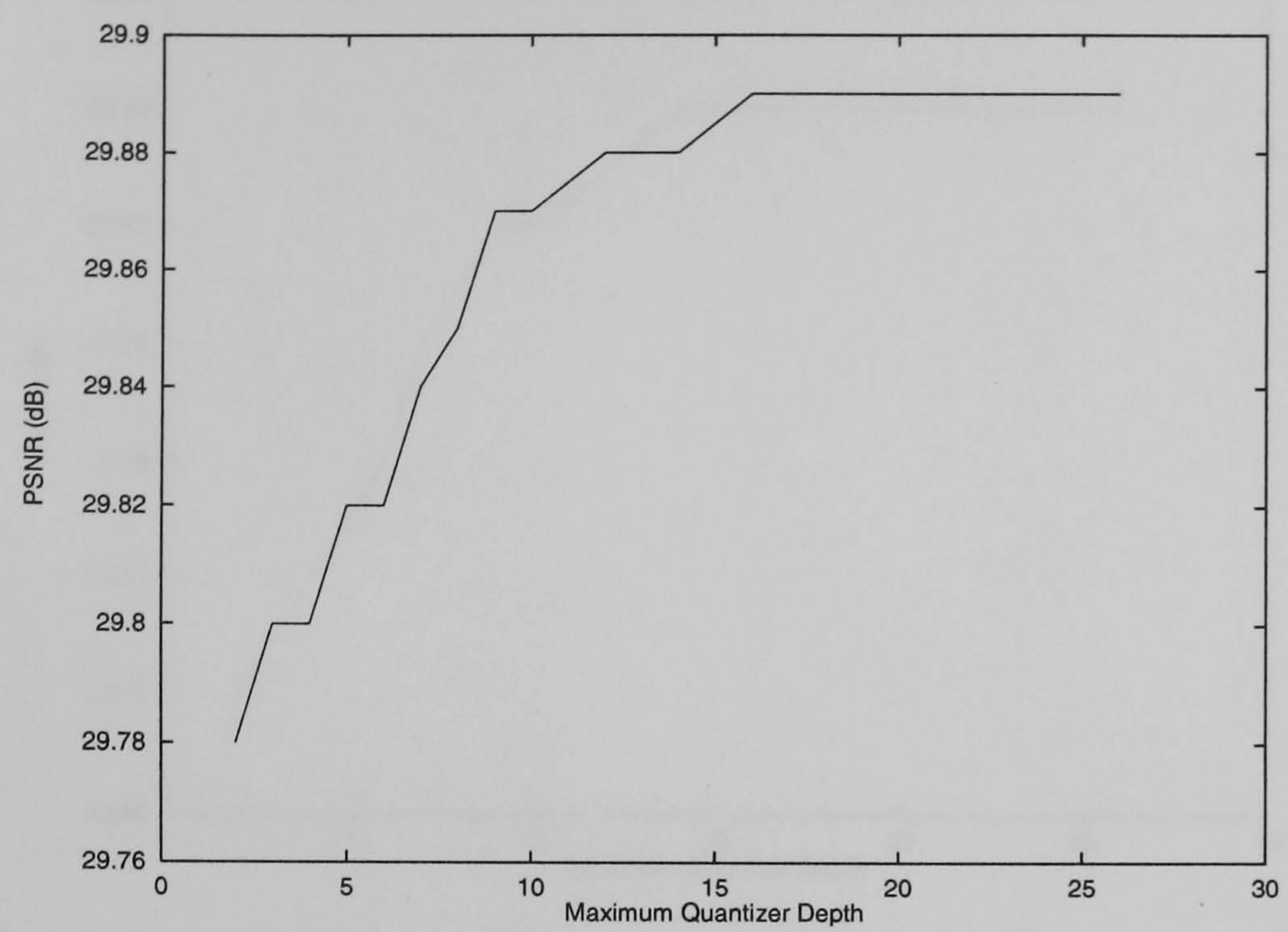


Figure 4.25: Effect of quantizer search depth on reconstruction error, *Lena* image.

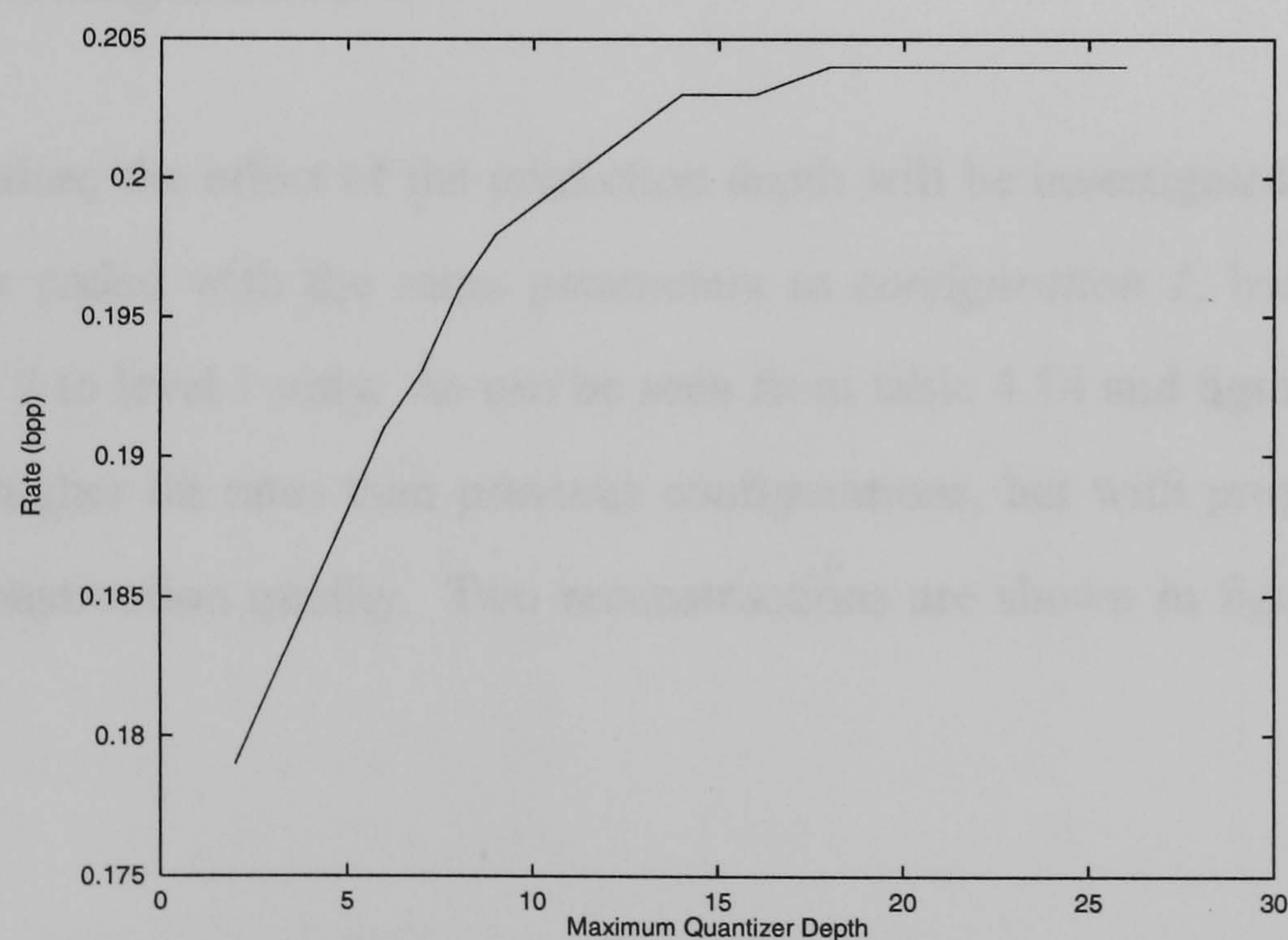


Figure 4.26: Effect of quantizer search depth on bit rate, *Barbara* image.

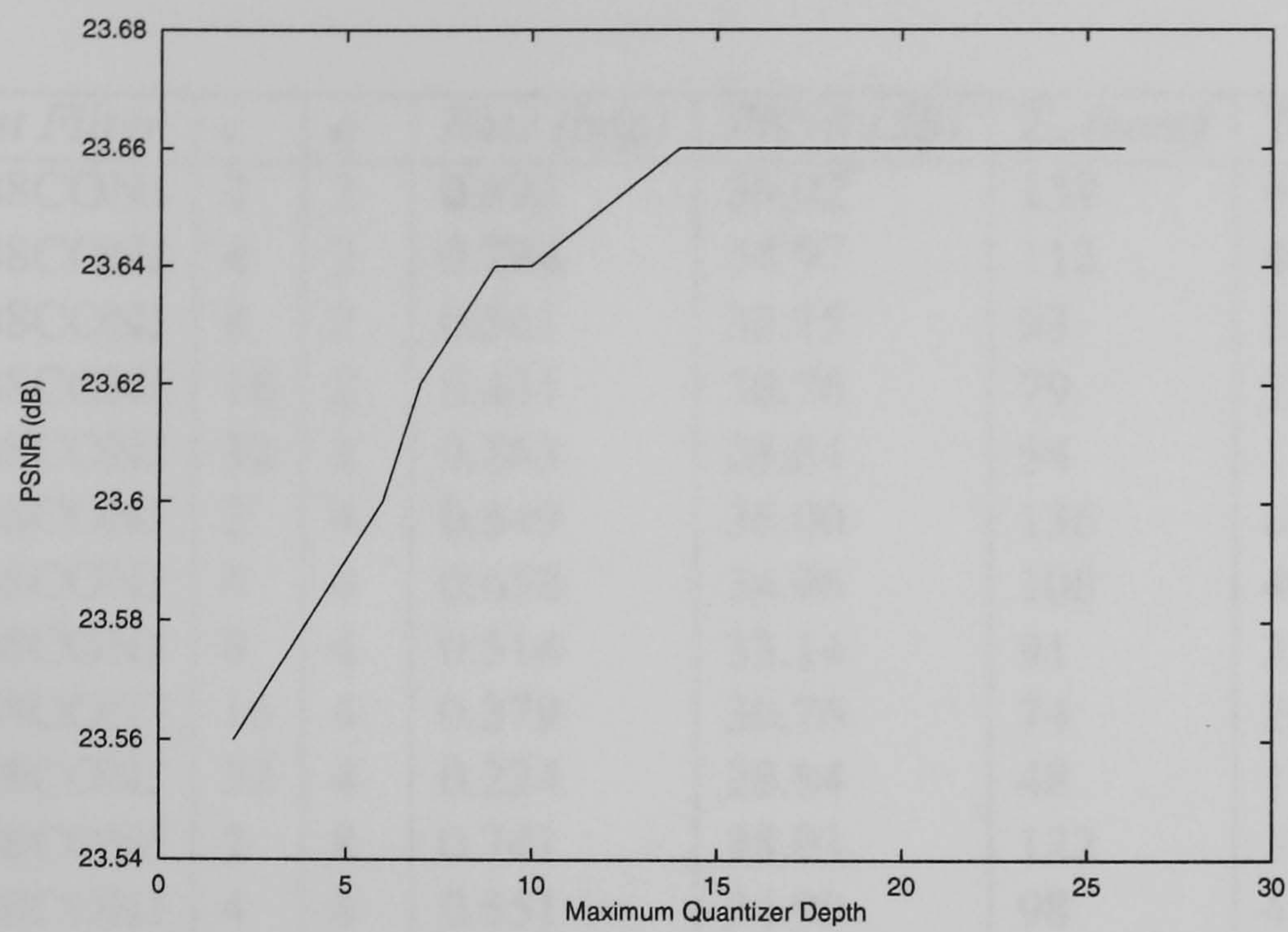


Figure 4.27: Effect of quantizer search depth on reconstruction error, *Barbara* image.

4.5.4 Configuration 4

In this section, the effect of the prediction depth will be investigated. The *Lena* image was coded with the same parameters as *configuration 1*, but predicting from level 2 to level 1 only. As can be seen from table 4.14 and figure 4.28, this produces higher bit rates than previous configurations, but with proportionately better reconstruction quality. Two reconstructions are shown in figures 4.29(a) and (b).

Wavelet Filter	*c*	*q*	*Rate (bpp)*	*PSNR (dB)*	T_e *(secs)*	T_d *(secs)*
DAUB8CONJ	2	2	0.892	36.02	139	65
DAUB8CONJ	4	2	0.704	34.97	112	43
DAUB8CONJ	8	2	0.561	33.15	93	32
DAUB8CONJ	16	2	0.431	30.76	79	23
DAUB8CONJ	32	2	0.263	28.84	54	11
DAUB8CONJ	2	4	0.849	36.00	136	60
DAUB8CONJ	4	4	0.658	34.96	108	42
DAUB8CONJ	8	4	0.514	33.14	91	32
DAUB8CONJ	16	4	0.379	30.76	74	21
DAUB8CONJ	32	4	0.224	28.84	48	11
DAUB8CONJ	2	8	0.741	35.93	122	58
DAUB8CONJ	4	8	0.551	34.90	98	46
DAUB8CONJ	8	8	0.411	33.11	85	31
DAUB8CONJ	16	8	0.288	30.74	70	21
DAUB8CONJ	32	8	0.167	28.84	46	10
DAUB8CONJ	2	16	0.629	35.78	114	57
DAUB8CONJ	4	16	0.448	34.79	93	41
DAUB8CONJ	8	16	0.314	33.04	79	30
DAUB8CONJ	16	16	0.211	30.71	64	21
DAUB8CONJ	32	16	0.126	28.82	44	10
DAUB8CONJ	2	32	0.582	35.56	109	60
DAUB8CONJ	4	32	0.406	34.63	91	39
DAUB8CONJ	8	32	0.273	32.93	76	32
DAUB8CONJ	16	32	0.175	30.65	62	20
DAUB8CONJ	32	32	0.106	28.80	43	9
DAUB8CONJ	2	64	0.559	35.28	117	55
DAUB8CONJ	4	64	0.384	34.42	88	40
DAUB8CONJ	8	64	0.254	32.80	77	31
DAUB8CONJ	16	64	0.158	30.58	61	20
DAUB8CONJ	32	64	0.096	28.77	58	12

Table 4.14: Results for Coder Configuration 4, *Lena* image

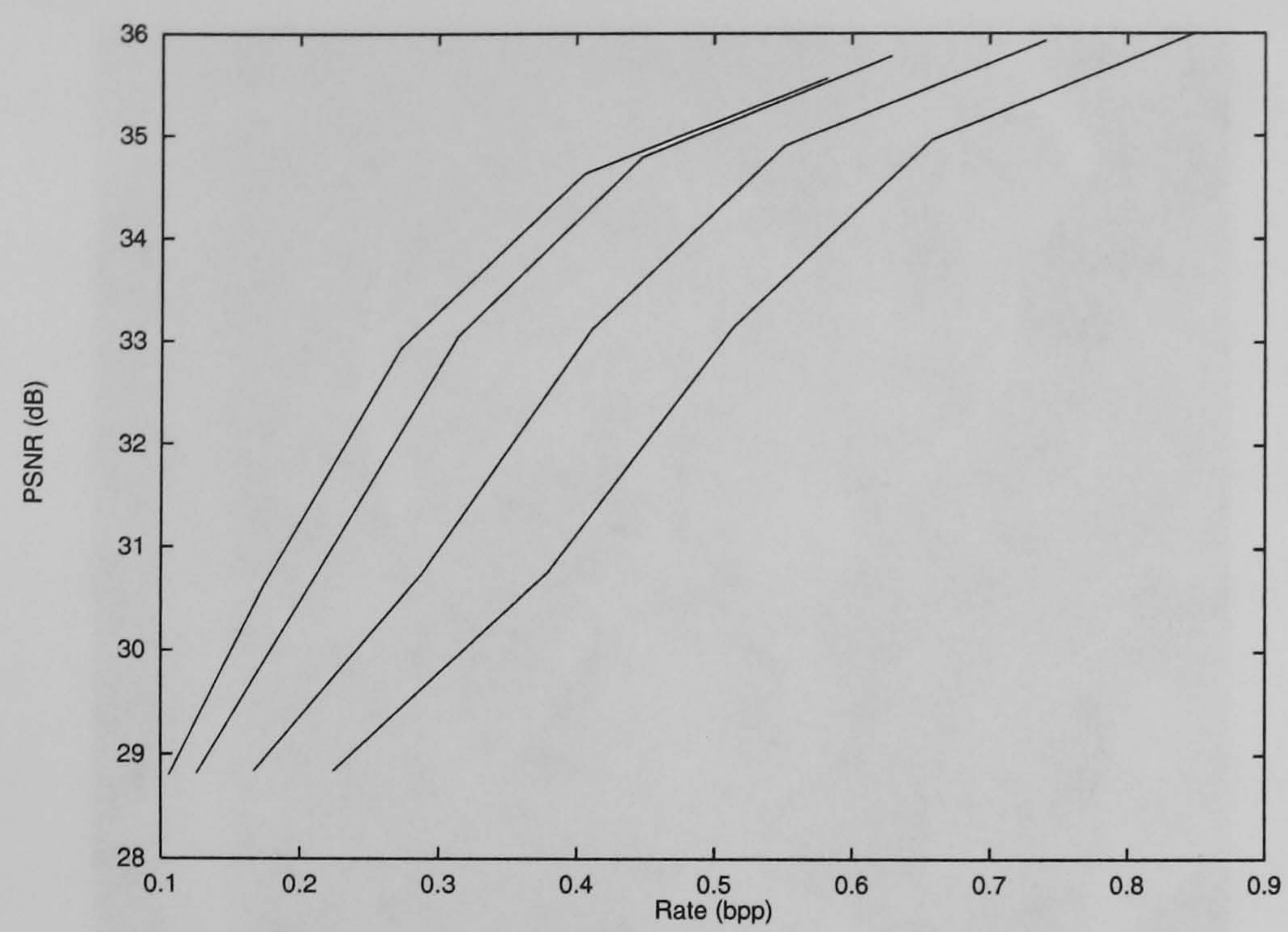


Figure 4.28: Results for coder configuration 4, *Lena* image.

4.6 Discussion

The results presented here exercise the coder on a variety of images at a variety of bit rates using a variety of wavelet filters. Consider the images shown in figure 4.16(a)-(d). These show the *Lena* image coded at approximately $0.10bpp$ using various wavelet filters. From inspection, it is apparent that (a) and (b) are, subjectively, more pleasing reconstructions. The $DAUB8CONJ$ and $BIORTH610$ filters appear to preserve strong edges better, for example on the girl's shoulder . However, the ringing artifacts are also more pronounced, for example at the top of the mirror. The $BIORTH1810$ filter whose reconstruction is shown in figure 4.16(c) appears to 'spread out' the edges which, while decreasing the appearance of the ringing artifacts, reduces the overall subjective reconstruction

(a)

(b)

Figure 4.29: Reconstructions from coder configuration 4 *Lena* image, (a) $c = 4, q = 2, 0.704bpp, 34.97dB$ (b) $c = 8, q = 16, 0.314bpp, 33.04dB$

quality. In the author's opinion, the $DAUB8CONJ$ filter produces the most visually pleasing results, and this filter is used to generate the rest of the results.

To attempt to understand how the coder weights the perceptual importance of different features, consider the reconstructions in figures 4.17, 4.19 and 4.21. At the higher bit rates (above $0.1bpp$), the only significant artifacts are present around very high frequency textures, for example the clothing in figure 4.21(b), and some ringing around diagonal edges, for example the masts in figure 4.19(b). As the wavelet quantizer step size becomes more coarse, the accuracy of the representation of the lower frequency wavelet coefficients is reduced. Since the VQ trees are initially loaded from this approximation, the efficacy of the prediction is reduced. At lower bit rates, the VQ scale factor is also heavily quantized. This reduces the ability of the vector quantizer to represent features present in the higher resolution wavelet bands accurately. Because of this, edges in the reconstructions at low bit rates appear to ring badly, as in figure 4.17(d). Also, it becomes difficult to represent textures with such large quantizer granularity. For example, the table cloth in figure 4.21(d) is reconstructed as an almost smooth surface apart from one or two patches of texture. These patches appear to be where the cloth is folded (thereby providing an edge) in the original image. They are, however, quite disturbing visually.

The vector quantizer search depth has little effect on reconstruction quality. When the scale factor quantizer step size is large, there are relatively few non-zero coefficients and so relatively few blocks are actually encoded. Taking this into account, the difference in reconstruction between using a full depth search and using a restricted search is significant, giving an increase of approximately $0.10dB$ for an increase of $0.012bpp$. There is almost no difference in the encoding and decoding times. In order to see the effect that the vector quantizer has on reconstructed image quality, compare figures 4.17 (c) and (d). The wavelet quantizer granularity is identical in both figures. In figure 4.17(c) the VQ scale factor quantizer granularity is relatively small, while in (d) the step size is so large that all coefficients quantize to zero, effectively disabling the VQ portion of the coder. From inspection, it is easy to see that the VQ portion of the coder does increase subjective quality of strong features, for example around the eyes and the edge of the mirror.

The wavelet transforms used in this work perform wrap around at the image edges. From inspection of the reconstructed images at the edges, it is apparent that this may not be the most suitable choice. For example, in figure 4.17(c), there is heavy ringing along the top edge and the top of the left hand edge. This appears to be where there is a large difference between the grey scale values on the opposing limits of the image. This creates an edge when wrapped around and consequently produces the disturbing ringing artifacts. A possible simple solution to this would be to employ periodic extension or repetition of the data instead of wrapping around.

4.6.1 Comparison With Other Published Works

In this section, the quality of the results of the coder presented in this chapter are compared with published work on other coding methods. The most common measure of image fidelity is the peak signal to noise ratio (PSNR), based on the mean square error. This fully quantifiable measure, however, does not take into account the properties of the human visual system. As such, an image with a higher PSNR (i.e. better quality numerically) may contain artifacts which are more visually disturbing. Fisher [Fis95] goes as far as to say

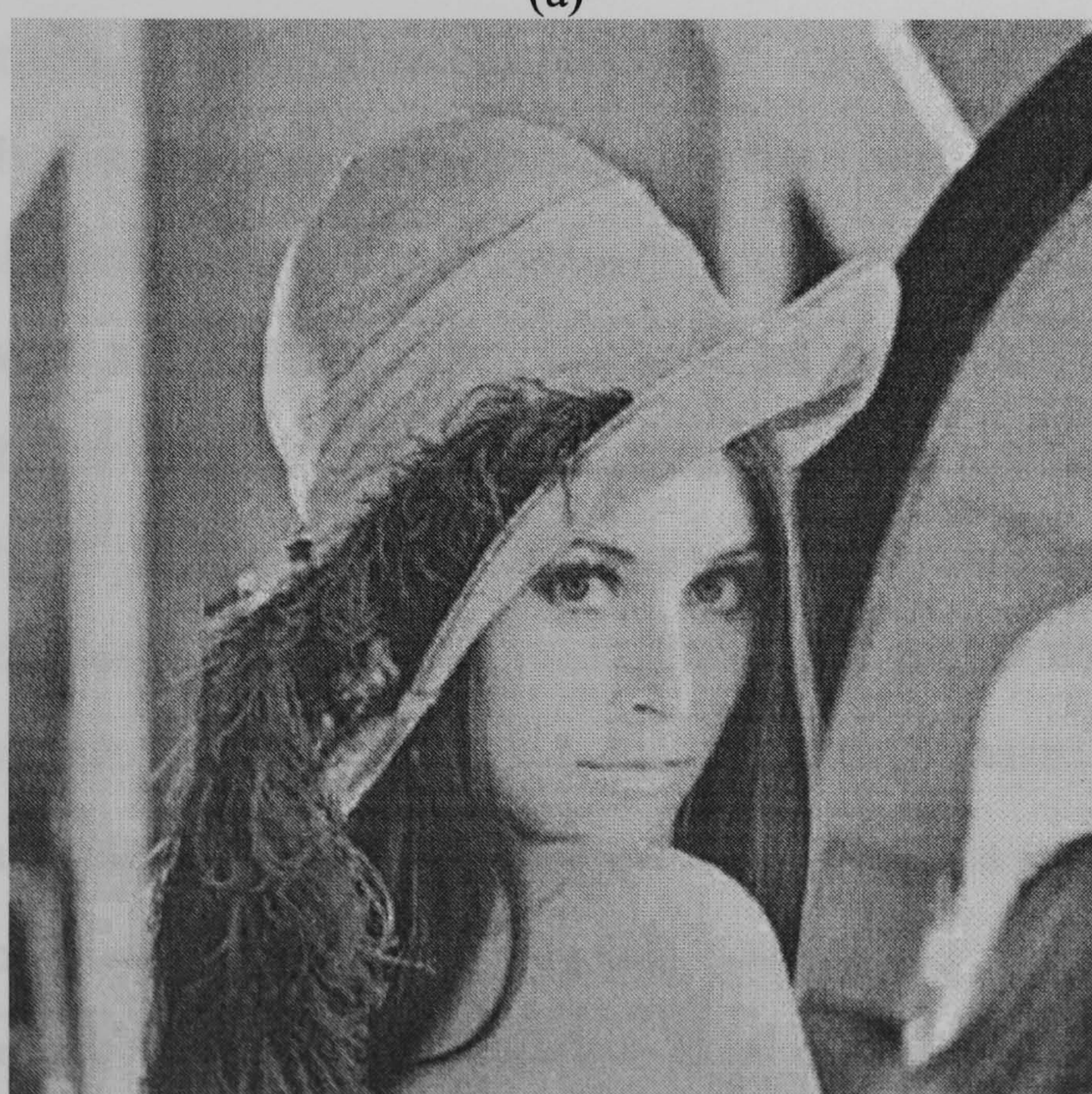
> The results should be taken with a grain, or more aptly a large mountain, of salt . . . PSNR is not a measure of perceived quality.

Also, it is known that there are several versions of each well-known test image and it is impossible to tell which version is used in other published material.

Figure 4.30(a) show a JPEG reconstruction of the *Lena* image at $0.13bpp$, with a PSNR of $24.86dB$ [3]. Figure 4.16 (a) shows the same image coded at 0.106 bpp, with a PSNR of $28.90dB$, over $4dB$ better. However, figure 4.30(b) shows a JPEG reconstruction at $0.31bpp$, with a PSNR of $33.26dB$. The coder presented here, using *configuration 4* will produce a reconstruction error of $33.04dB$ in $0.314bpp$ (figure 4.29(b)), comparable to the JPEG coder. Arguably, there is little to distinguish the subjective quality of the two higher bit rate reconstructions.

[3] This was produced by the JPEG coder in the common *xv* programme, set to produce the lowest bit-rate.

(a)



(b)

Figure 4.30: JPEG reconstructions of *Lena* at different bit rates. (a) $0.13bpp$, $24.86dB$ (b) $0.31bpp$, $33.26dB$.

Table 4.15 compares the reconstruction results for the coder presented here with currently published works. It is apparent from the table that the algorithm presented here performs as well as or better than all the comparative methods at low bit rates. Shapiro's EZW coder [Sha93] remains the benchmark for all image coding algorithms at low rates. The coder presented here matches Shapiro's coder closely at the lower bit rates for the *Lena* image. The *Barbara* image used by Shapiro appears to be slightly different from the one used here, although the results are included for completeness. The *Boats* and *Barbara* images are more difficult to code than the *Lena* image because of the greater degree of texture and prevalence of high frequency artifacts.

The coder presented here does not work well at high bit rates and the encoding of the lower frequency bands of the wavelet transform may be the cause. On the other hand, at low bit rates the reconstructions do not appear to suffer from the highly visually disturbing artifacts usually associated with block based coders. The artifacts that are introduced at lower bit rates appear to be caused by the ringing of the wavelet filters around edges, since altering the wavelet filters alters the character of the artifacts. A post-processing algorithm, such as an adaptation of the adaptive thresholding algorithm in [GLGO94], may well reduce these artifacts and increase the visual quality of the reconstructed images. It also seems likely that a more symmetric set of orientation bands, i.e. ones not based on a separable transform, may well be better adapted to human visual system properties. Design of such a set of filters is a highly complex task, however.

[4] The *Barbara* image used by Shapiro appears to be different to the one used here.

Image	*Presented Results*		*Published Results*		
	Rate	PSNR	Rate	PSNR	Method/Ref
Lena	0.106 Figure 4.16(a)	$28.90dB$	0.125	$30.23dB$	EZW [Sha93]
			≈ 0.11	$\approx 27dB$	Evolutionary Fractal [SR96]
			≈ 0.10	$\approx 26.5dB$	Evolutionary Fractal [SR96]
			≈ 0.13	$\approx 28.5dB$	Fractal/VQ [HMS96]
			0.192	$30.84dB$	Embedded DCT [LLK96]
			0.10	$28.1dB$	Fractal/HV Partition [FM95]
Lena	0.074 Figure 4.17(c)	$27.21dB$	0.0625	$27.54dB$	EZW [Sha93]
			0.08	$26.81dB$	Fractal/Wavelet [KMK95]
			≈ 0.07	$\approx 25dB$	Fisher Quadtree [Fis95]
Lena	0.037	$25.14dB$	0.03125	$25.38dB$	EZW [Sha93]
			0.041	$24.23dB$	Embedded DCT [LLK96]
Boats	0.175 Figure 4.19(b)	$27.39dB$	0.181	$27.54dB$	Embedded DCT [LLK96]
	0.063 Figure 4.19(d)	$24.98dB$	0.064	$24.26dB$	Embedded DCT [LLK96]
Barbara[4]	0.109 Figure 4.21(c)	$22.88dB$	0.0625	$23.10dB$	EZW [Sha93]
	0.040 Figure 4.21(d)	$21.84dB$	0.03125	$21.94dB$	EZW [Sha93]

Table 4.15: Comparison with other published works.

Chapter 5

Video Coding Using A Three Dimensional Wavelet Transform

5.1 Introduction to Video Coding

The subject of still image coding is nearing maturity in the research community. Video coding is still, comparatively speaking, in its youth, however. Figures 5.1 and 5.2 show two consecutive frames from the *Miss America* sequence. From inspection, the difference between the two frames is small. Indeed, this is a requirement for a sequence of still images to be interpreted by the human visual system as a moving video. Most, if not all, video coding schemes use some form of interframe coding. One of the earliest versions of interframe coding was that of Mounts [Mou69]. Each frame is examined and only elements which have changed sufficiently (with respect to a given threshold) between frames are transmitted. All other image elements are taken from the decoder's current buffer, giving a basic

Figure 5.1: Frame 40 from the Miss America Sequence

Figure 5.2: Frame 41 from the Miss America Sequence

form of prediction [Cla95]. In predictive coding, previously coded pixels are used as a *context* for encoding the current pixel. The predictor is analogous to the two dimensional case, except that temporal neighbours are also included. As an extension to this, transform domain predictors have also been used in video coding [Cla85]. In these, the current and previous frames are transformed and the prediction is applied to the transform coefficients. This has the advantage that, given the correct choice of transform and quantization, many of the coefficients will be zero, enabling simple entropy coding.

Many of the more modern video coding schemes try to predict the motion of visually important features. In this way, it is argued, the encoder can spend more bits on the visually important features of the sequence. Indeed, the encoder and decoder both have the same prediction algorithm to predict where features may move to in the next frame. If the prediction is correct, very little data need to be sent. However, if the prediction is wrong, information must be sent which will allow the decoder to correct its prediction. The predictions are usually based on some form of model. If the features or the motion do not fit the model, the prediction errors will be large and occur frequently. This will drive up the bit rate or reduce the quality of the reconstructed sequence. The most widely used prediction method in video coding algorithms is *motion compensation.* Motion compensation attempts to predict where a group of pixels (be they a feature or some geometric patch) will move to in the next frame. By sending such a motion vector, the decoder can just apply the motion vector to the block in frame n to produce the moved block in frame $n+1$. The motion vectors produced by this method are often a very limited subset of the possible motions. If the full range of motion was allowed, the information required to encode the parameters of each motion would probably outweigh the information saved in the prediction. As with any predictive system, motion compensation is accompanied by transmitting a difference signal, to reduce the error rate and prevent instability. For example, the current industry standard video coding algorithm, MPEG [LeG92] [Tek95], uses motion compensation. Motion compensation does have its problems, though. Motion vectors are usually limited to simple translations to reduce computation. Obviously, this simplification does not always fit the data, causing errors. The video coding research community is currently very active in improving motion compensation schemes. Multiscale (hierarchical) compensation and generalized compensation, where the motion is not so restricted, are both current research topics [SMP97]. A more robust method would perhaps not use such a simple model and would not, therefore, be limited to certain features and motions. There are the widely accepted concepts of *features*, *motion*, *direction* and *velocity*. Combinations of these concepts will determine whether something is required to be coded accurately or can be ignored. For example, a small feature moving very quickly will be largely ignored by the human visual system [Wat88]. Some form of transform which separates the image data into classes of *features* and *non-features* and which further separates the features into their respective velocities and directions would therefore provide an ideal base for a video compression algorithm.

5.2 The Three Dimensional Wavelet Transform and Video Coding

The separable two dimensional wavelet transform presented in chapter 2 is the basis of many recent still image coders. It would, therefore, be sensible to attempt to extend the theory to video coding, simply by adding an extra dimension to the data to account for the time domain. Formally, the three dimensional separable continuous wavelet transform is

$$\begin{aligned}&(W_\psi f)\left(a_x, a_y, a_t, b_x, b_y, b_t\right) = \\ &|a_x|^{\frac{1}{2}}|a_y|^{\frac{1}{2}}|a_t|^{\frac{1}{2}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y,t)\psi\left(\tfrac{x-b_x}{a_x}\right)\psi\left(\tfrac{y-b_y}{a_y}\right)\psi\left(\tfrac{t-b_t}{a_t}\right)dxdydt\end{aligned} \tag{5.1}$$

In practice, since the transform is Cartesian separable, a three dimensional discrete transform can be realised by performing the one dimensional transform along the third axis, after applying the two-dimensional transform to each image slice. Figure 5.3 shows a representation of a three dimensional wavelet decomposition. The temporal decomposition produces temporal frequency bands, analogous to the spatial frequency bands. These temporal frequency bands represent objects moving at different velocities. Therefore, each spatio-temporal band contains information about features of differing sizes and orientations which are moving at differing speeds in differing directions.

In the two dimensional case, the wavelet coefficient bands are numbered simply $1, \ldots, n$ from the highest to the lowest scale. For the three dimensional wavelet transform, the labelling scheme for spatio-temporal frequency bands is a simple extension of this. The temporal bands are also labelled $1, \ldots, n$, with 1 being the

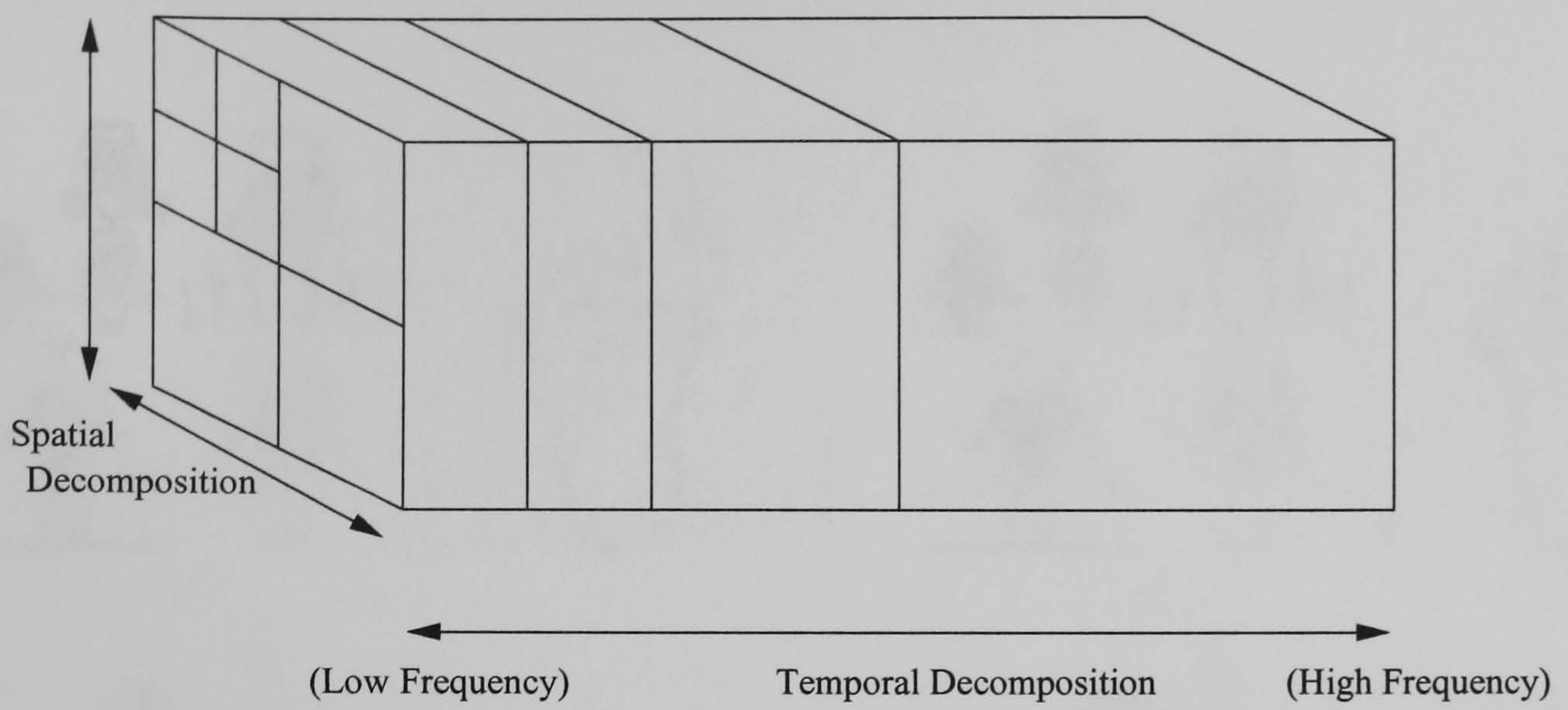


Figure 5.3: A representation of a three dimensional wavelet transform

highest temporal frequency. A spatio-temporal band is then labelled (a, b) where a is the spatial wavelet level and b is the temporal wavelet level. Figures 5.4, 5.5 and 5.6 show some of the temporal low pass, lowest frequency highpass and the next lowest frequency highpass, from the *Miss America* sequence where the transform was applied to 3 levels in both spatial and temporal domains. The spatial highpass has been removed for clarity of detail and only the magnitudes of the coefficients are shown. It is simple to observe from these slices that the temporal frequency bands do contain what would be expected. The temporal lowpass band contains features which are relatively stationary and the other two bands contain the features which move, such as the eyes and mouth.

Each band has a specific spatial frequency and orientation as defined by the spatial transform applied to each image slice and a temporal frequency as defined by the temporal transform applied to the whole sequence. The human visual system

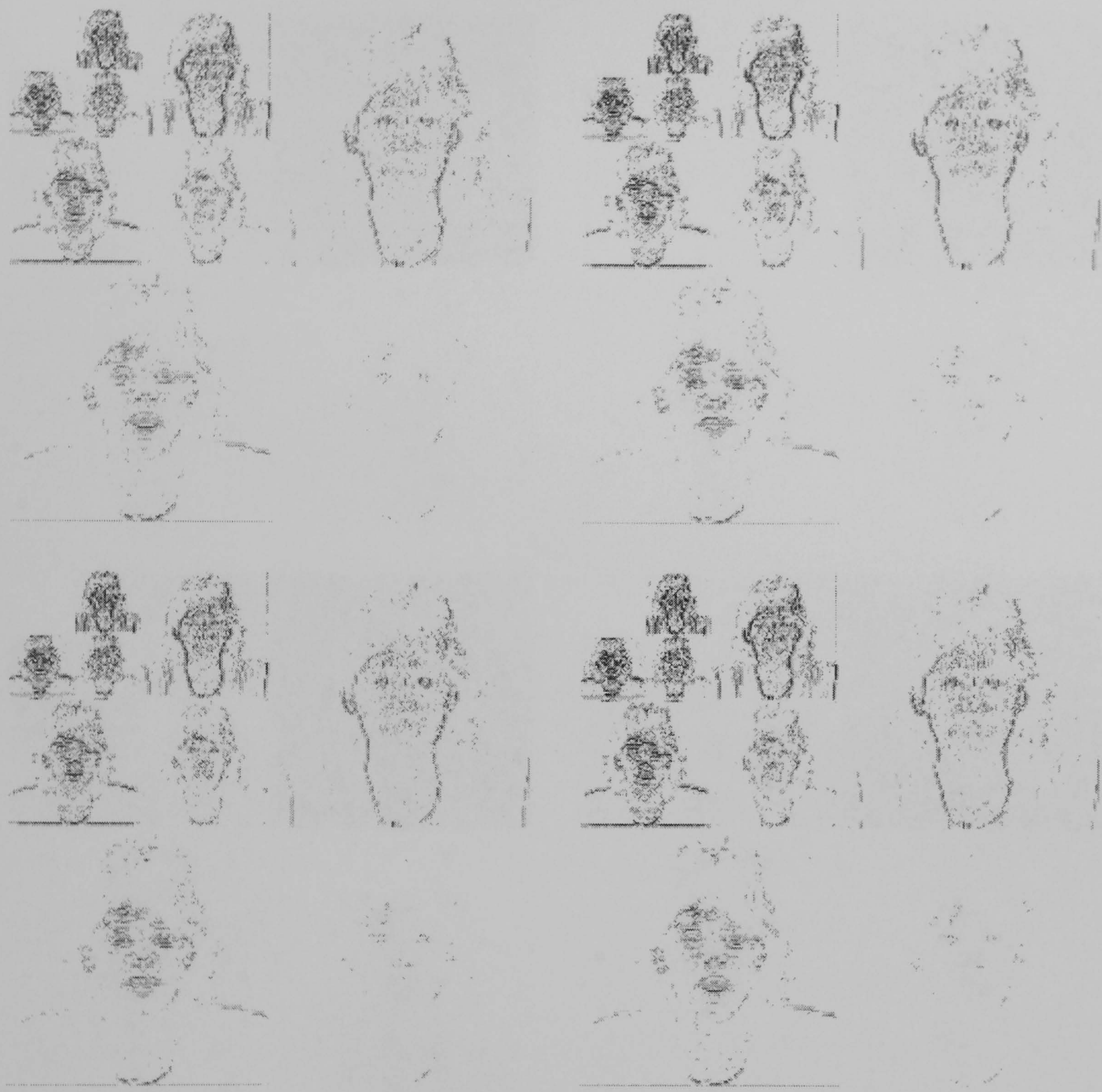


Figure 5.4: Four slices from the temporal lowpass band of Miss America transform (spatial LP removed). The magnitudes only are shown.

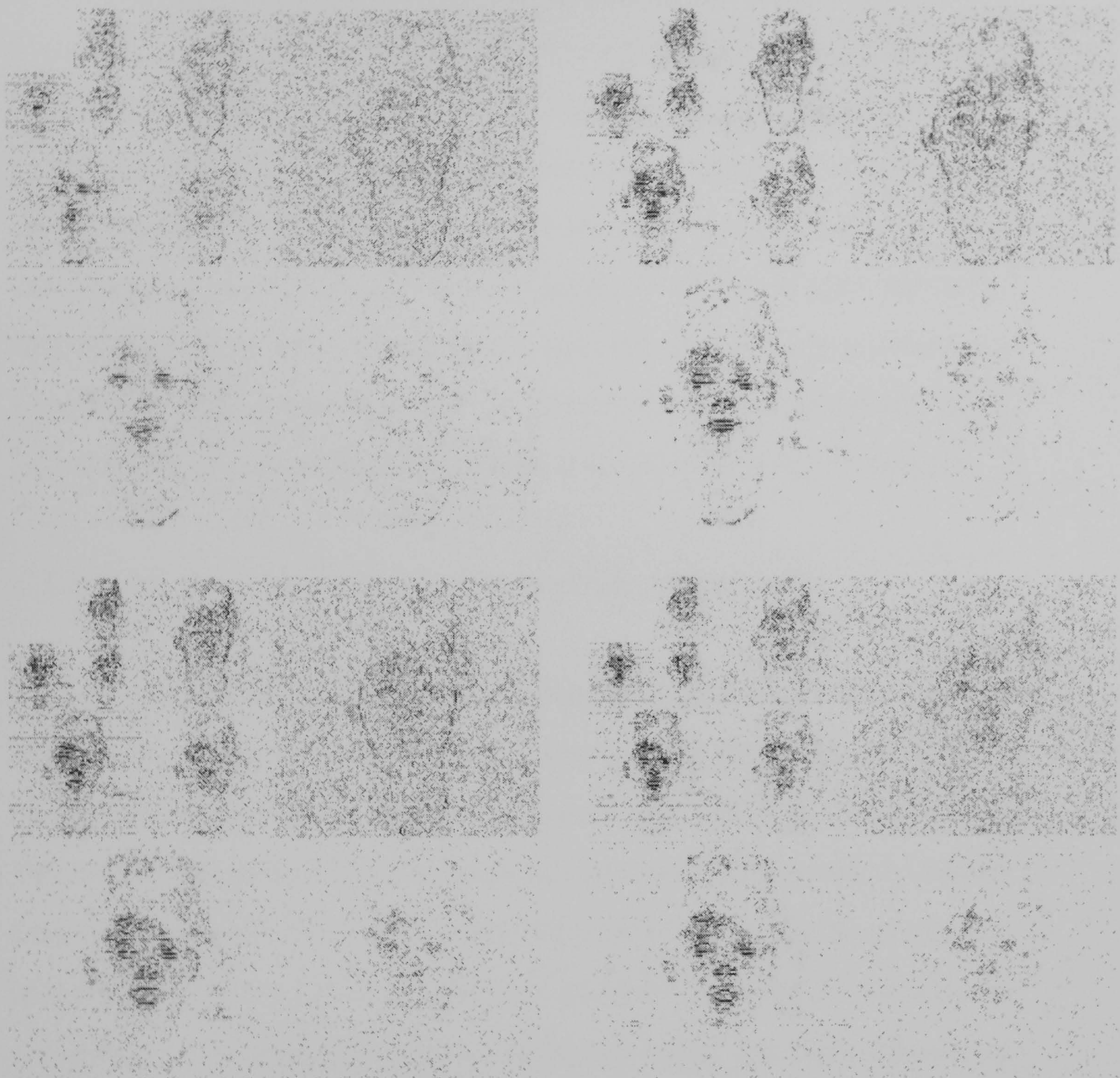


Figure 5.5: Four slices from temporal level 3 of Miss America transform (spatial LP removed). The magnitudes only are shown.

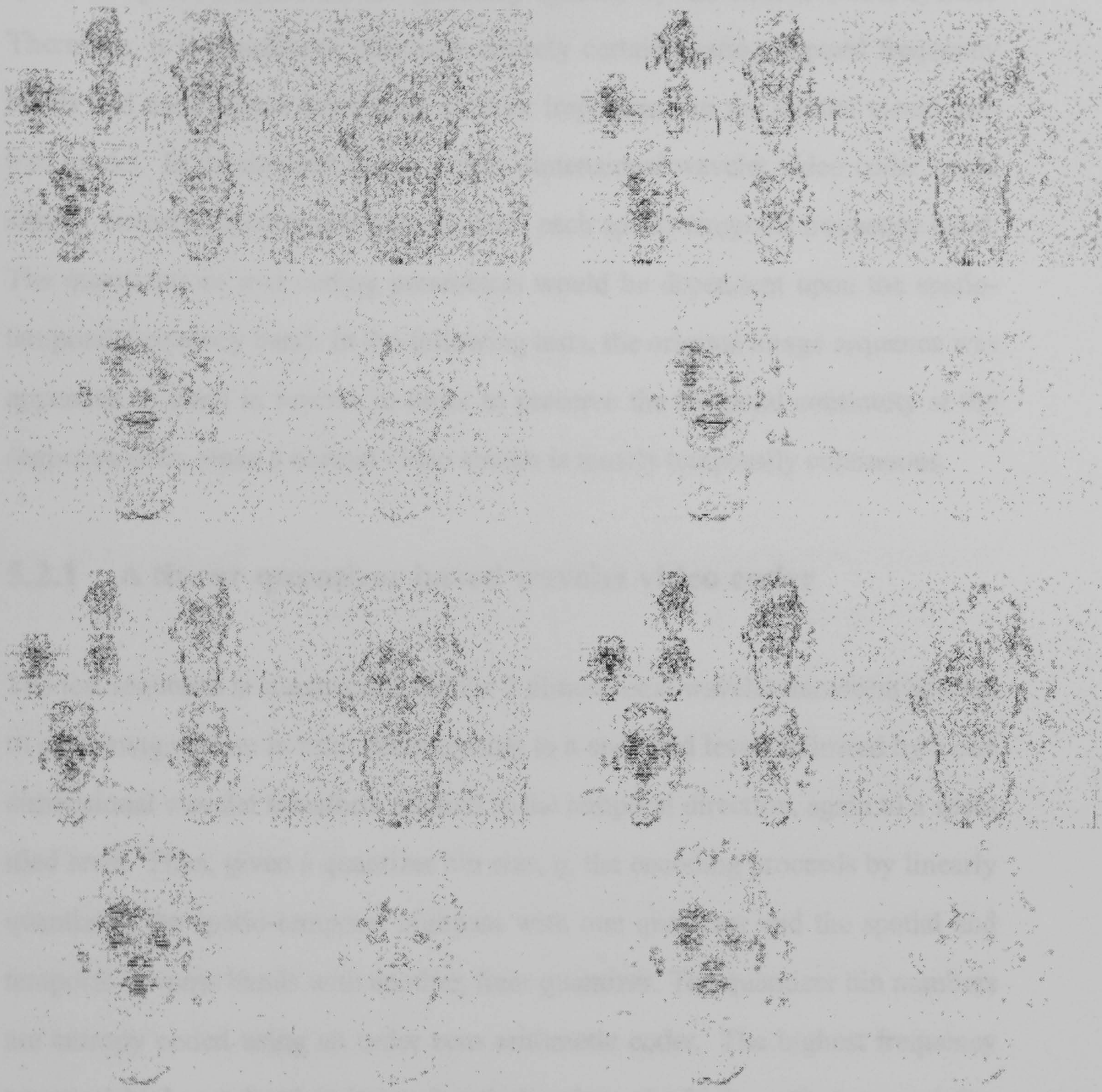


Figure 5.6: Four slices from temporal level 2 of Miss America transform (spatial LP removed). The magnitudes only are shown.

is only responsive to certain spatio-temporal frequencies [Wat88]. Indeed, high spatio-temporal frequencies are generally ignored by the human visual system. Therefore, it is possible to eliminate entirely certain spatio-temporal frequency bands and code others depending on their importance in the human visual system model. In its simplest form, a three dimensional wavelet video coder could simply linearly quantize and entropy code each spatio-temporal frequency band. The quantizations and coding parameters would be dependent upon the spatio-temporal frequency band. In the following tests, the original image sequence was appended to itself in reverse in order to preserve the temporal continuity at the sequence ends, since a normal video stream is mostly temporally continuous.

5.2.1 A linear quantizer based wavelet video coder

The test sequence is constructed and the 2 dimensional wavelet transform applied to each image frame in turn, decomposing to a specified level, followed by a one dimensional wavelet transform applied in the temporal direction, again to a specified level. Then, given a quantizer bin size, q, the encoding proceeds by linearly quantizing the spatio-temporal highpass with one quantizer and the spatial and temporal lowpass bands with another, finer quantizer. The quantizer bin numbers are entropy coded using an order zero arithmetic coder. The highest frequency temporal highpass level is ignored entirely, since the high spatio-temporal frequency data in this band are largely ignored by the human visual system. This extremely simple method produces some remarkably good results, detailed next.

Filter	*Rate Value*	*Average PSNR*	*Bits per Pixel*
DAUB4	16	36.75	0.080
DAUB4	32	35.21	0.045
DAUB4	64	33.39	0.025
DAUB4	128	31.14	0.014
DAUB8	16	37.20	0.078
DAUB8	32	35.61	0.044
DAUB8	64	33.83	0.024
DAUB8	128	31.46	0.014
DAUB16SYM	16	37.51	0.077
DAUB16SYM	32	35.91	0.043
DAUB16SYM	64	34.07	0.024
DAUB16SYM	128	31.61	0.014

Table 5.1: Results for linear quantizer based wavelet video coder on the Miss America test sequence

The linear quantizer based coder was tested on the *Miss America* sequence, and the filters are taken from those detailed in chapter 4. The spatial and temporal wavelet transforms were both applied to three levels. The results produced by runnning the coder on the *Miss America* sequence using various quantizer bin sizes and wavelet filters are shown in table 5.1 and figure 5.7. A smaller set of results are presented for the *Table Tennis* sequence in table 5.2 and figure 5.8. As a guide to transmission rates, assuming 256×256 pixel frames at 15 frames per second, 0.01 bits per pixel is equivalent to approximately $9.8kbps$ while 0.045 bits per pixel is approximately $44kbps$.

To show how the linear quantizer coder performs over individual frames in a sequence, figure 5.9 shows the individual frame peak signal to noise ratio for the Miss America sequence coded using the $DAUB8$ filter and a rate value of 64.

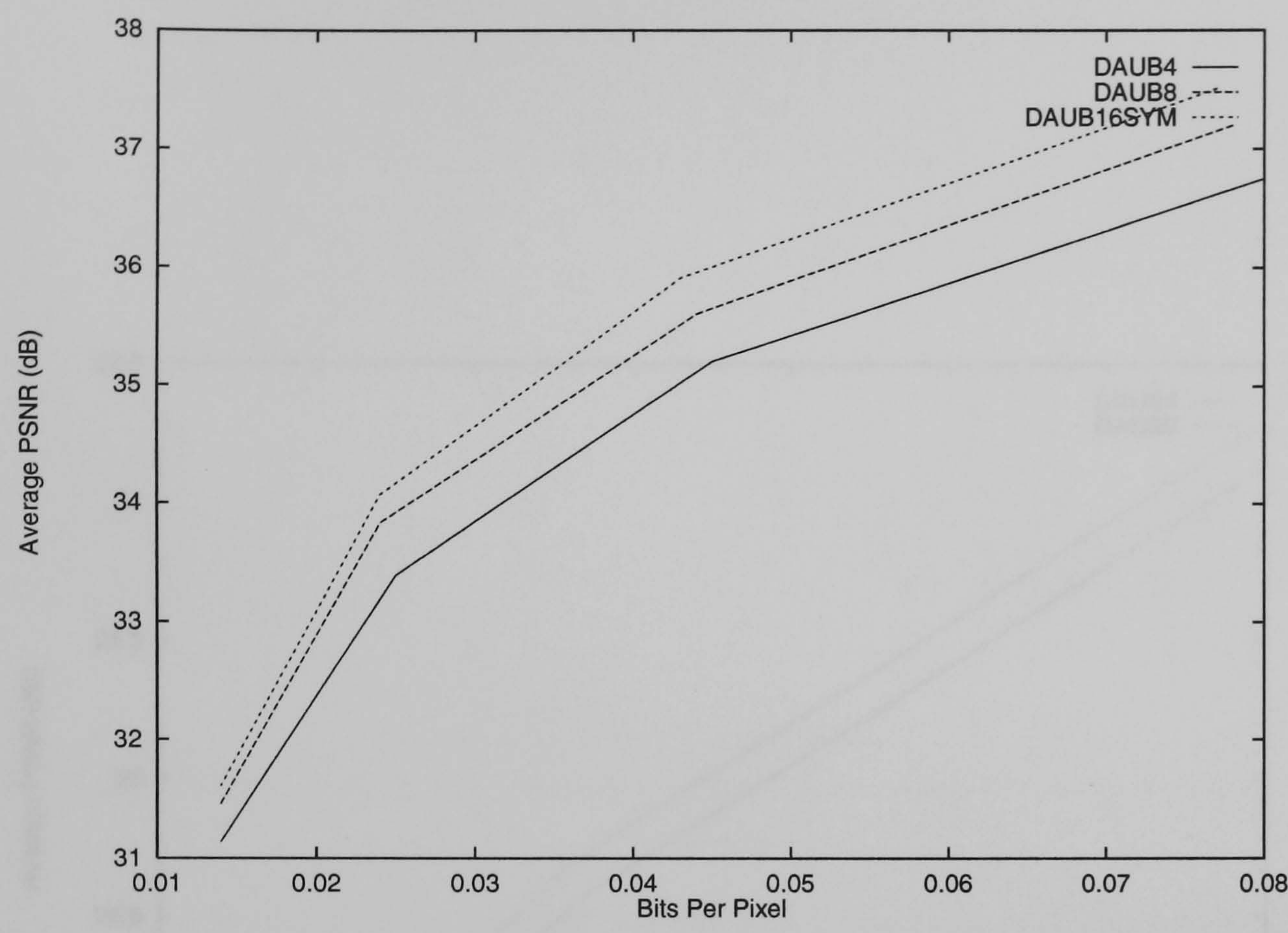


Figure 5.7: Results for linear quantizer based wavelet video coder on the Miss America test sequence

Filter	*Rate Value*	*Average PSNR*	*Bits per Pixel*
DAUB4	16	27.13	0.160
DAUB4	32	25.57	0.073
DAUB4	64	24.50	0.034
DAUB8	16	27.27	0.159
DAUB8	32	25.70	0.072
DAUB8	64	24.63	0.034

Table 5.2: Results for linear quantizer based wavelet video coder on the Table Tennis test sequence

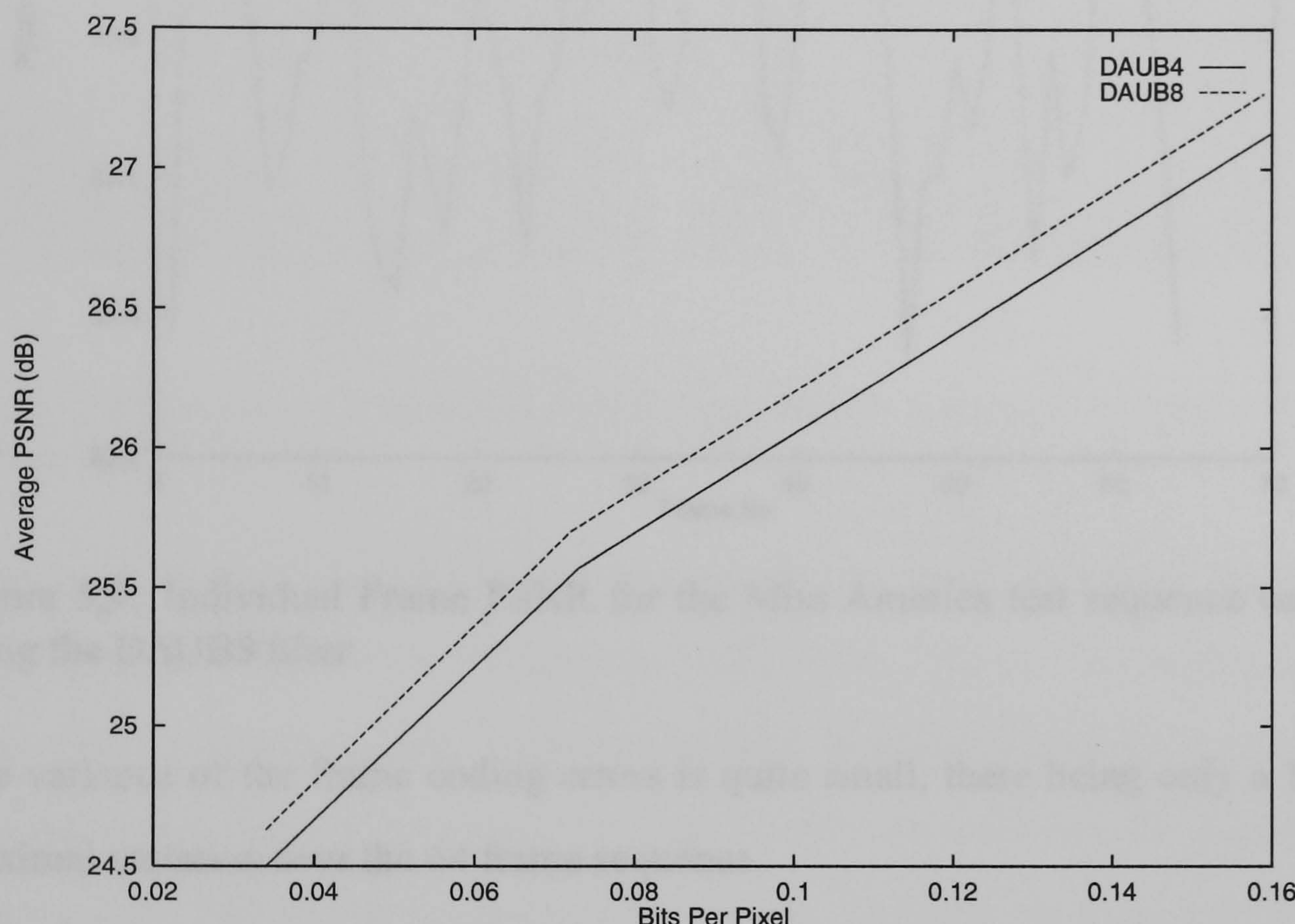


Figure 5.8: Results for linear quantizer based wavelet video coder on the Table Tennis test sequence

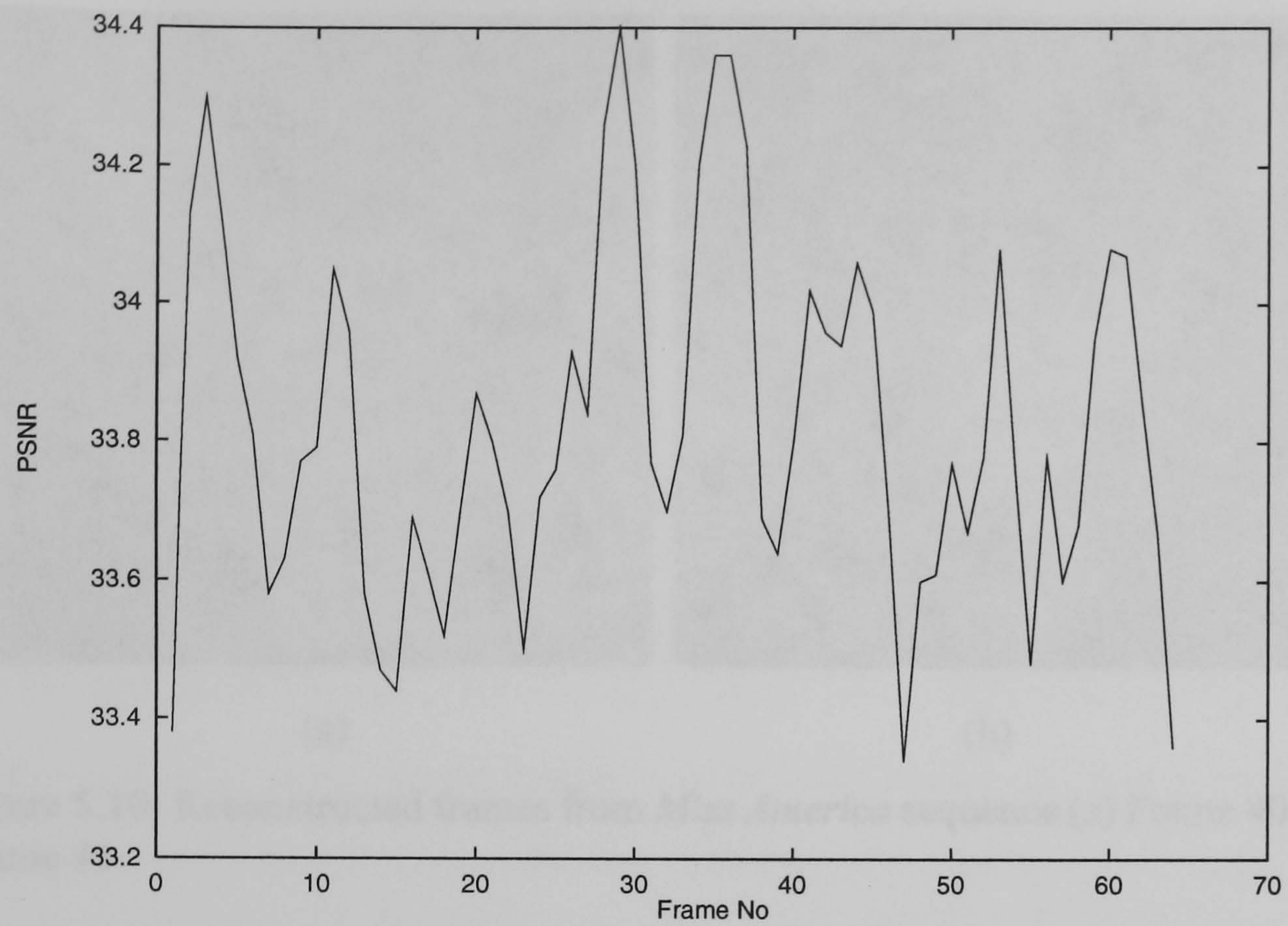


Figure 5.9: Individual Frame PSNR for the Miss America test sequence coded using the DAUB8 filter

The variance of the frame coding errors is quite small, there being only a $1dB$ maximal variation over the 64 frame sequence.

5.3 Properties and Pitfalls of the linear quantizer method

The linear quantizer video coder is simple, but does not adequately take into account visually important features. For example, the background of the *Table Tennis* sequence is textured. Because the background hardly moves, the related

(a) (b)

Figure 5.10: Reconstructed frames from *Miss America* sequence (a) Frame 40, (b) Frame 41

wavelet coefficients are in the spatio-temporal lowpass band. As such, a relatively large number of bits are used coding the background. This can unnecessarily increase the bit rate for a given subjective quality. In order to keep the overall bit rate down, the temporal and spatial highpass bands are more heavily quantized, leading to artifacts in the reconstructed sequence. Consider the frames shown in figures 5.10(a) and 5.10(b), taken from the *Miss America* sequence coded using the linear quantizer based coder at 0.025 bits per pixel, using the $DAUB4$ filter.

These frames occur in the sequence when the speaker's lips change direction. A change in direction can be thought of as a *temporal edge*. As discussed in chapter 4, wavelet coders have a tendency to produce ringing artifacts at edges at low bit rates. The speaker's lips ring badly on frame 40, which is the extremum of the lip movement and the system recovers on frame 41. A method that accounts for more of the structure and features of the underlying transform is required.

5.4 A VQ based Video Coding Algorithm

It is widely known in the image coding community that a well designed vector quantizer should outperform a linear quantizer [LBG80]. A two dimensional vector quantizer encodes patches of an image. By extension, a three dimensional vector quantizer will encode patches of an image in a certain time interval. That is, the vector quantization representative vectors will consist of features that pass across the image patch during the time window. This has obvious similarities to the decomposition provided by the three dimensional wavelet transform. To begin, the video image sequence is decomposed by the spatio-temporal wavelet transform. In all the tests that follow, the transform was applied to three spatial and three temporal levels. The highest temporal frequency band is not coded. As stated previously, the high spatio-temporal frequency data present in this band is relatively perceptually unimportant to the human visual system. The temporal lowpass band and all spatial lowpass bands are quantized with a uniform, linear scalar quantizer with a relatively small bin size. The spatial lowpass contains continuous tone sections of image and the temporal lowpass contains features which, relatively speaking, do not move during the sequence. Scalar quantizing this data with a relatively fine quantizer allows for a good overall impression of the sequence to be produced with no detail. The detail from the sequence is eminently suitable for vector quantization. In order to produce a generalised coder, a general purpose vector quantization codebook must be created. To produce this codebook, a variety of image sequences was scanned and appropriately sized cuboids of data extracted from the appropriate wavelet coefficients. In these tests, the vectors were $4 \times 4 \times 4$ pixels and were accumulated from spatio-temporal wavelet level $(3, 3)$. As in the still image coder presented in section 4.4, two tree structured vector quantizers are used: one for the horizontally and vertically oriented data and one for the diagonally oriented data. The trees contain vectors which are stored in a canonical orientation. The time-reverse, per-pixel negation and negated time reverse of any vectors are also stored. Time reversal allows for objects moving in either direction and per-pixel negation is used to ensure that the generated codebook has a zero vector. If the energy of any particular block extracted is below a given threshold, it is treated as noise and does not contribute to the vector accumulation. Different thresholds are used for the H-V and D trees.

Given a vector quantizer codebook, spatio-temporal bands $(3, 3)$,$(2, 2)$,$(3, 2)$ and $(1, 3)$ are vector quantized and the VQ indices are entropy coded using an order zero arithmetic coder. Recall from section 2.4.3 that the discrete wavelet transform is directly equivalent to a multiresolution analysis. Recall also, equation 2.18, repeated here for convenience. Assume that $V_m \subset L^2(\Re)$ is a multiresolution analysis. Then

$$f \in V_m \Leftrightarrow f(2\cdot) \in V_{m-1}. \tag{5.2}$$

Moving to the frequency domain , this equation states that if a function $f \in V_m$

has a frequency of ω, then the functions contained in the subspace V_{m-1} will have frequency 2ω. Define v to be a velocity. Then, a translation at velocity v of a complex exponential is simply

$$e^{\imath\omega(x-vt)} = e^{\imath\omega x}e^{-\imath\omega vt} \tag{5.3}$$

from which

$$v = \frac{\omega_t}{\omega_s}. \tag{5.4}$$

So, for a given spatial band n and temporal band t, the velocity of the features in spatio-temporal band (n, t) will be twice that of features in $(n - 1, t)$. Now, features present in spatio-temporal band $(2, 3)$ will be similar to those present in spatio-temporal band $(3, 3)$ except that they will be moving twice as fast. In order to account for the velocity difference, the quantizer vectors are dyadically downsampled in the temporal direction. Then the downsampled codebooks are used to encode spatio-temporal bands $(2, 3)$ and $(1, 2)$. Again, a simple, order zero arithmetic coder acts as the entropy coder. Figure 5.11 shows a block diagram of the coder.

In all the following tests, the H-V codebook contains 256 vectors and the diagonal codebook 64. The subsampled codebooks contain the same number of vectors as the original codebooks. These values are fairly arbitrary but investigation showed that these produced better results than other values in the range 64 to 1024 vectors per codebook. Since the VQ codebook is fixed, the rate control comes about from the variation of the quantization of the temporal lowpass bands.

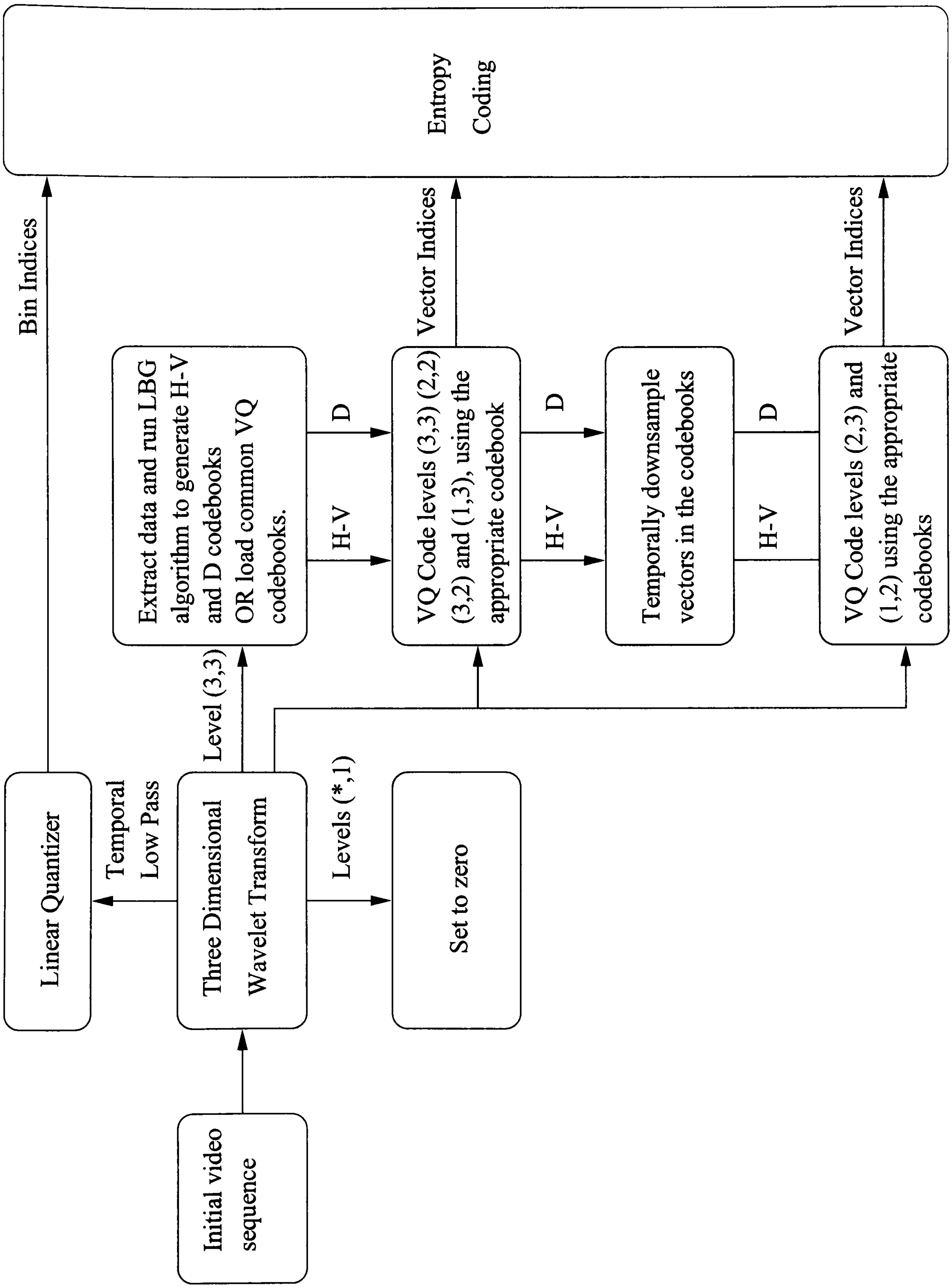


Figure 5.11: Vector Quantizer Based Coder Block Diagram

Filter	*Rate Value*	*Average PSNR*	*Bits per Pixel*
DAUB8	16	38.09	0.099
DAUB8	32	36.77	0.059
DAUB8	64	35.37	0.036
DAUB8	128	33.76	0.022
DAUB8	256	31.55	0.014
DAUB4	16	37.76	0.103
DAUB4	32	36.41	0.061
DAUB4	64	35.04	0.037
DAUB4	128	33.32	0.022
DAUB4	256	31.21	0.014
DAUB16SYM	16	38.36	0.097
DAUB16SYM	32	37.06	0.058
DAUB16SYM	64	35.63	0.035
DAUB16SYM	128	33.96	0.022
DAUB16SYM	256	31.71	0.014

Table 5.3: Results for vector quantizer based wavelet video coder on the Miss America test sequence

5.5 Results

The vector quantizer video coder was run on a variety of image sequences, using a variety of wavelet filters and set to produce a variety of bit rates. The results are presented in this section. Note that there is an inherent difficulty in selecting frames from a sequence for presentation. The human visual system has certain spatio-temporal response characteristics which affect subjective quality. The temporal component of sequences cannot be reproduced by still image frames, and so the quality of a reconstruction may be perceived differently depending on whether it is presented as part of a sequence or a still image frame.

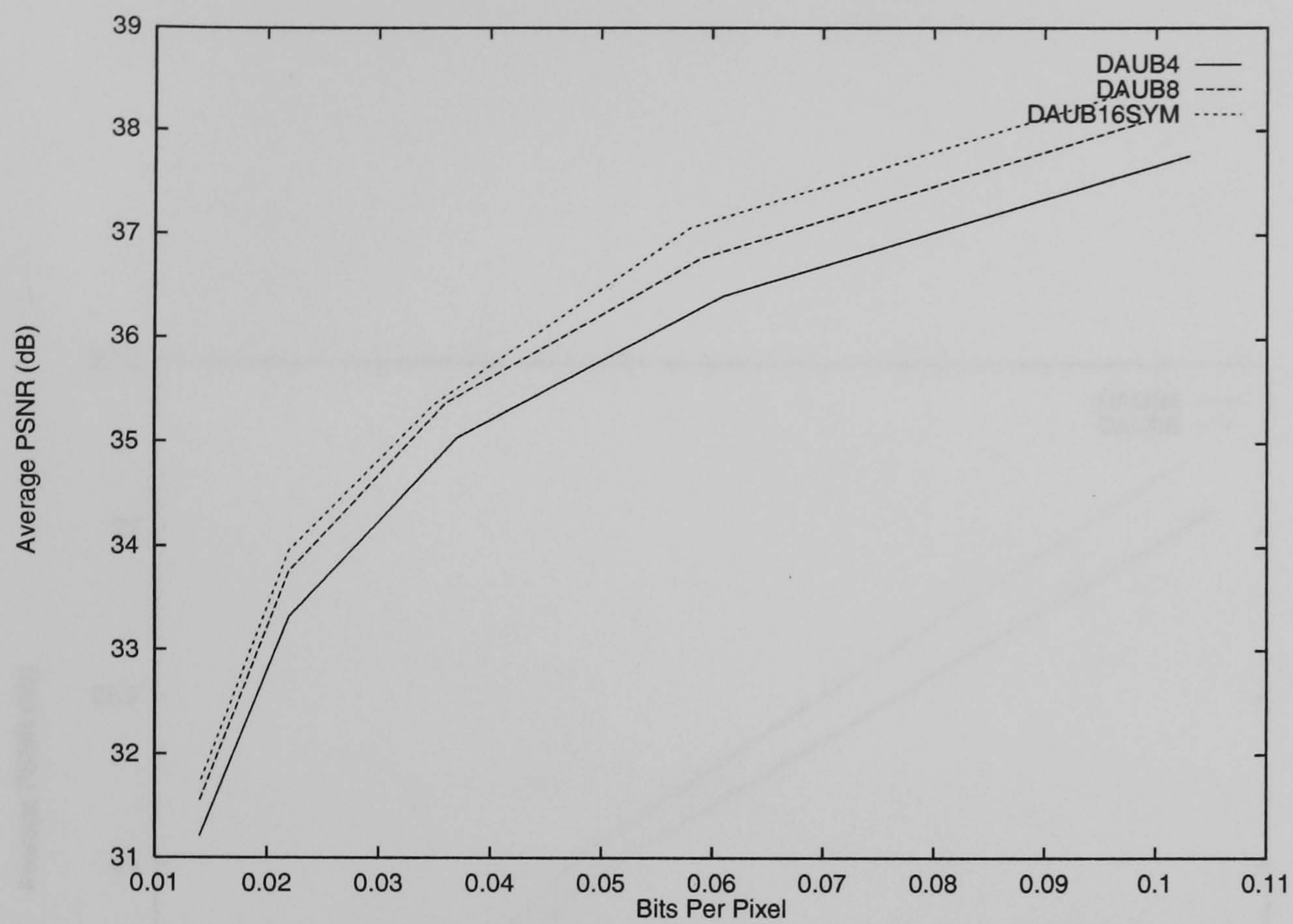


Figure 5.12: Compression of the Miss America sequence using various filters to produce various bit rates

Filter	*Rate Value*	*Average PSNR*	*Bits per Pixel*
DAUB4	16	27.06	0.230
DAUB4	32	25.96	0.118
DAUB4	64	25.02	0.059
DAUB8	16	27.21	0.225
DAUB8	32	26.04	0.117
DAUB8	64	25.06	0.058

Table 5.4: Results for vector quantizer based wavelet video coder on the Table Tennis test sequence

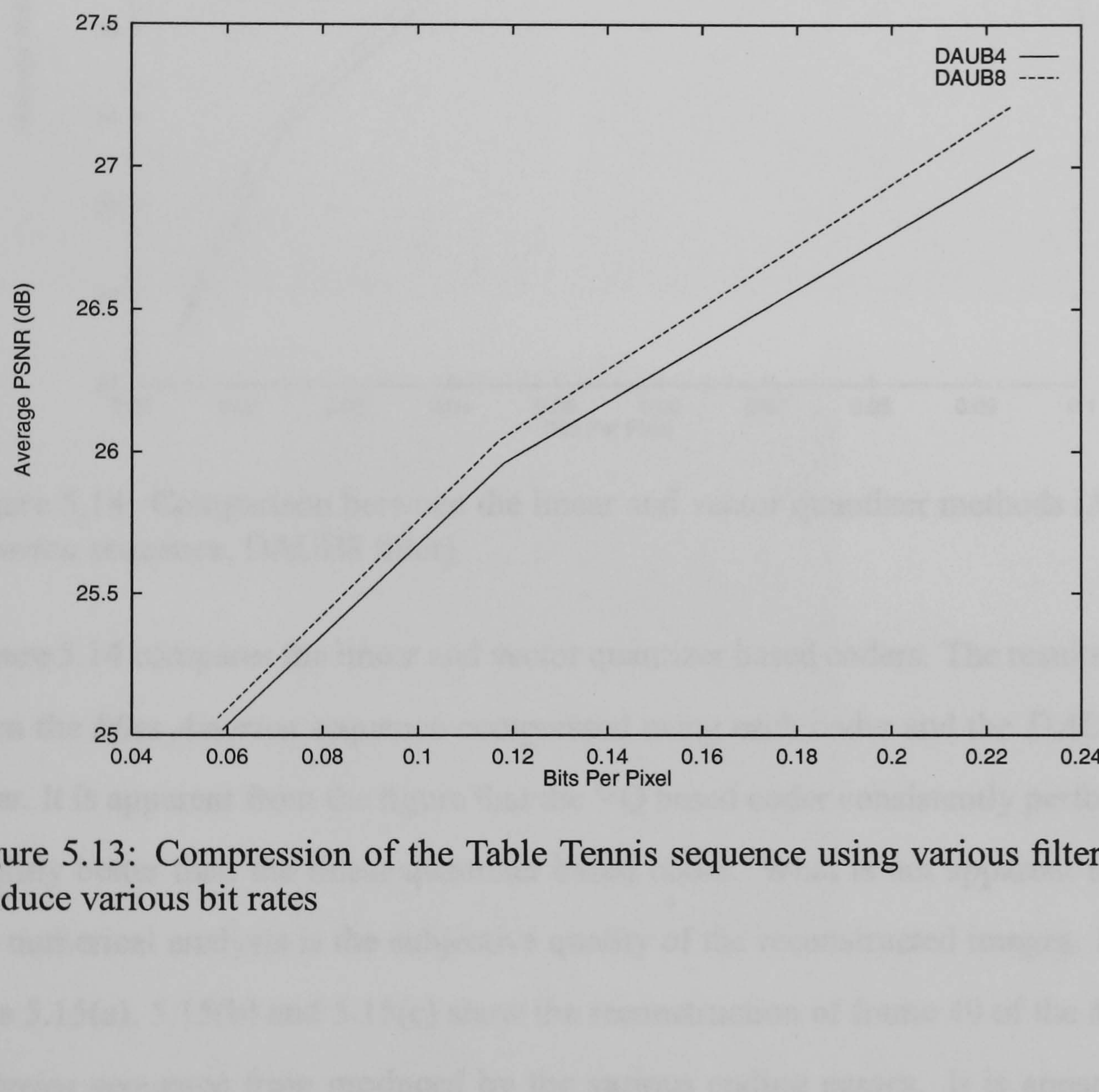


Figure 5.13: Compression of the Table Tennis sequence using various filters to produce various bit rates

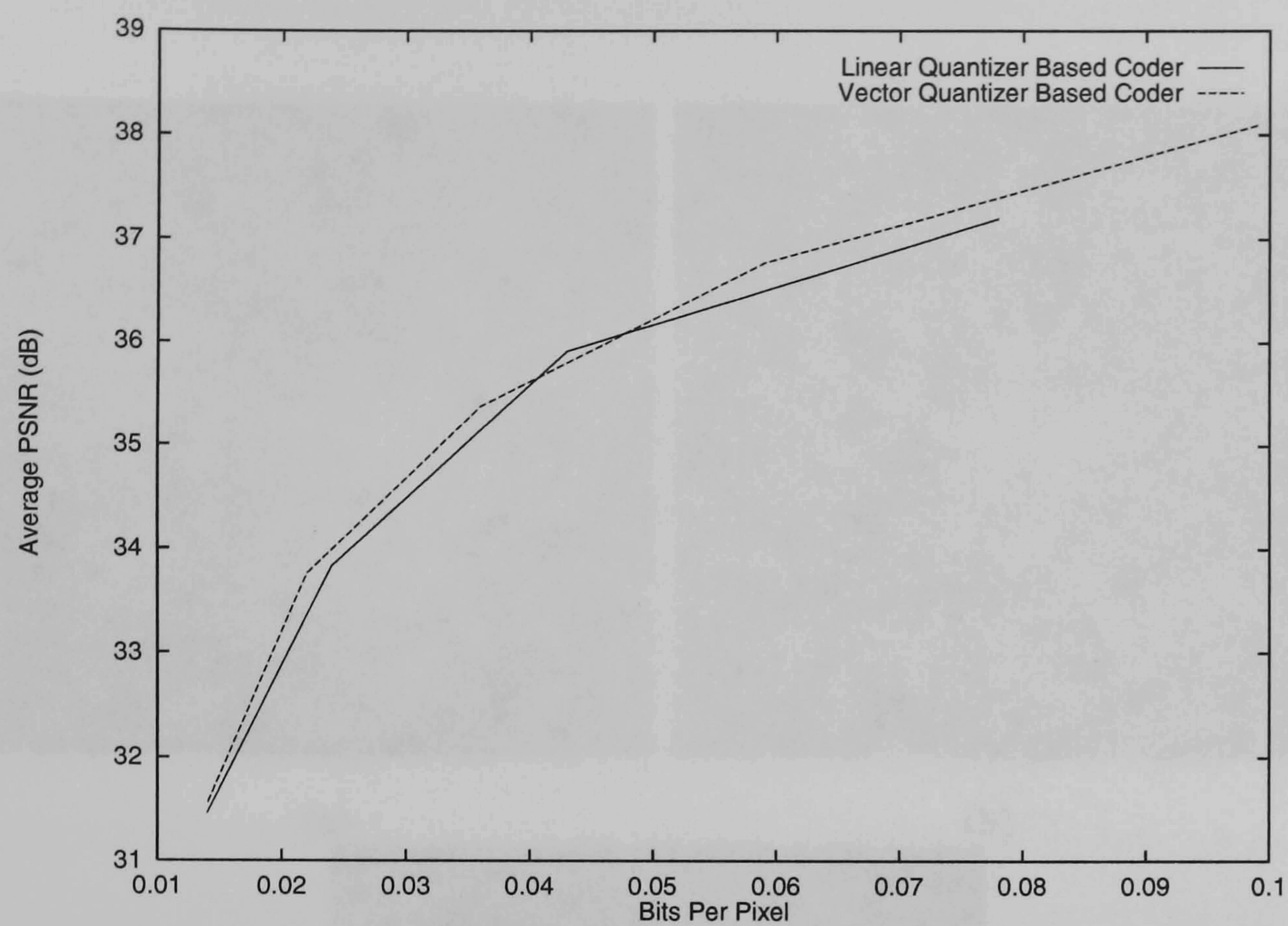
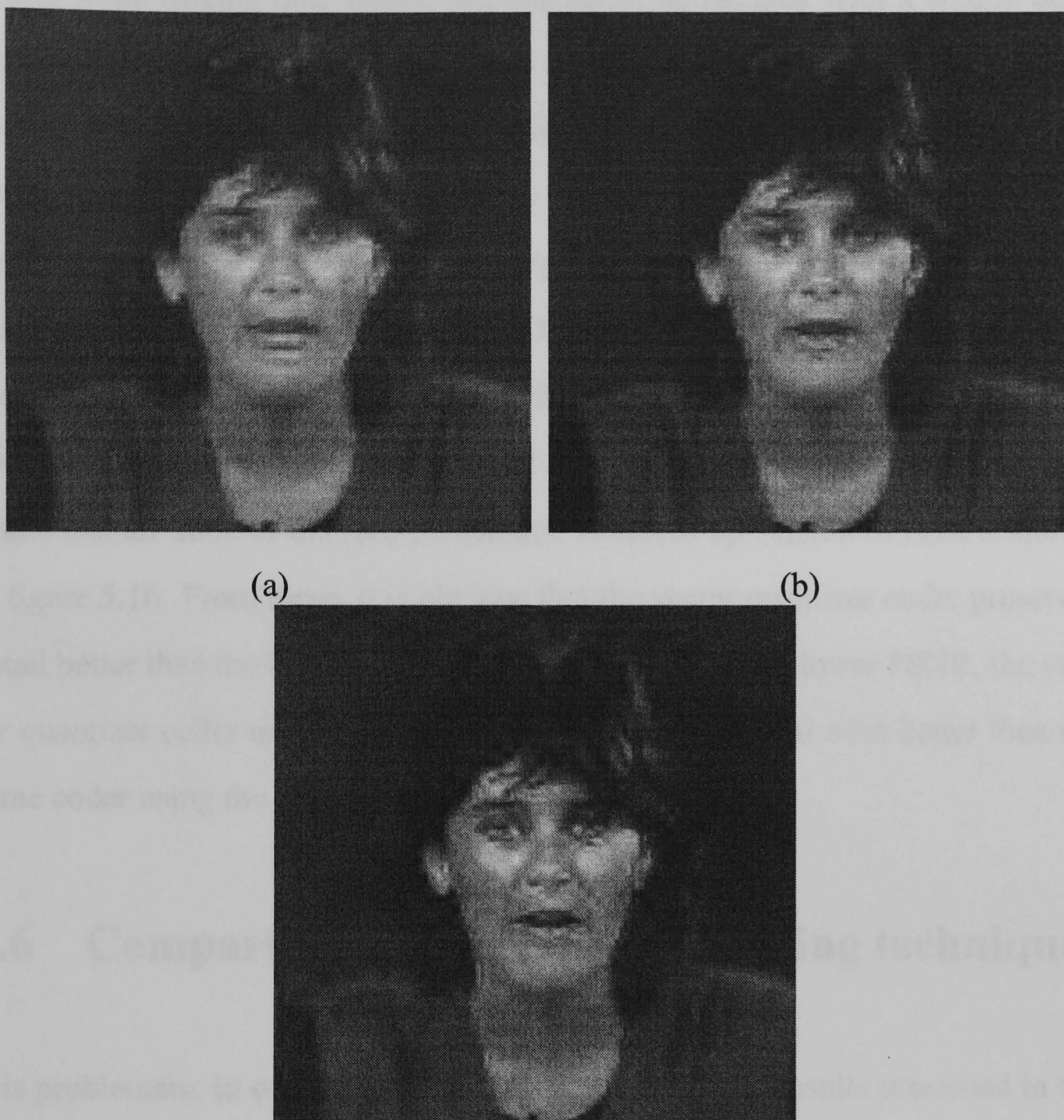


Figure 5.14: Comparison between the linear and vector quantizer methods (*Miss America* sequence, DAUB8 filter)

Figure 5.14 compares the linear and vector quantizer based coders. The results are from the *Miss America* sequence compressed using each coder and the $DAUB8$ filter. It is apparent from the figure that the VQ based coder consistently performs slightly better than the linear quantizer based coder. What is not apparent from the numerical analysis is the subjective quality of the reconstructed images. Figures 5.15(a), 5.15(b) and 5.15(c) show the reconstruction of frame 40 of the *Miss America* sequence from produced by the various coding passes. It is apparent, from inspection, that the vector quantizer approach produces reconstructions with more detail than the linear quantizer based coder. There are fewer large errors in



(a) (b)

(c)

Figure 5.15: Reconstructed frame 40 from *Miss America* sequence (a) Linear Quantizer, DAUB4 @ 0.025bpp, PSNR=34.01dB, (b) Vector Quantizer, DAUB4 @ 0.022bpp, PSNR=34.26dB, (c) Vector Quantizer, DAUB8 @ 0.022bpp, PSNR=33.66dB

the VQ reconstruction and any artifacts appear to have a higher frequency. There is still some ringing near edges, but this could be reduced with a simple post-processing algorithm. Since the artifacts are of a higher frequency, peak signal to noise ratio is not a particularly useful measure of subjective quality. Indeed, the vector quantizer, by its very nature, will attempt to preserve more detail than its scalar counterpart. This is easily seen by comparing the eyes in figures 5.15(a), 5.15(b) and 5.15(c). The VQ based coder produces better defined eyes in both cases, and a much better defined mouth in figure 5.15(c) in spite of the change in the filter used. To aid inspection, the left eye has been extracted from the original frame and all three of the reconstructions. A scaled up version of each is shown in figure 5.16. From these, it is obvious that the vector quantizer coder preserves detail better than the linear quantizer coder. Also, despite a lower PSNR, the vector quantizer coder using the $DAUB8$ filter preserves detail even better than the same coder using the $DAUB4$ filter.

5.6 Comparison with other video coding techniques

It is problematic to compare video coding schemes from results presented in the literature. This is because research groups seem to use different measures, sometimes not fully specifying them. For example, some results are presented as bits per second, without specifying frame size or rate. In order to provide a basis for comparison, where full details are not provided, it is assumed that the sequence is in CIF format [Cla95]. All data from the relevant research papers are converted

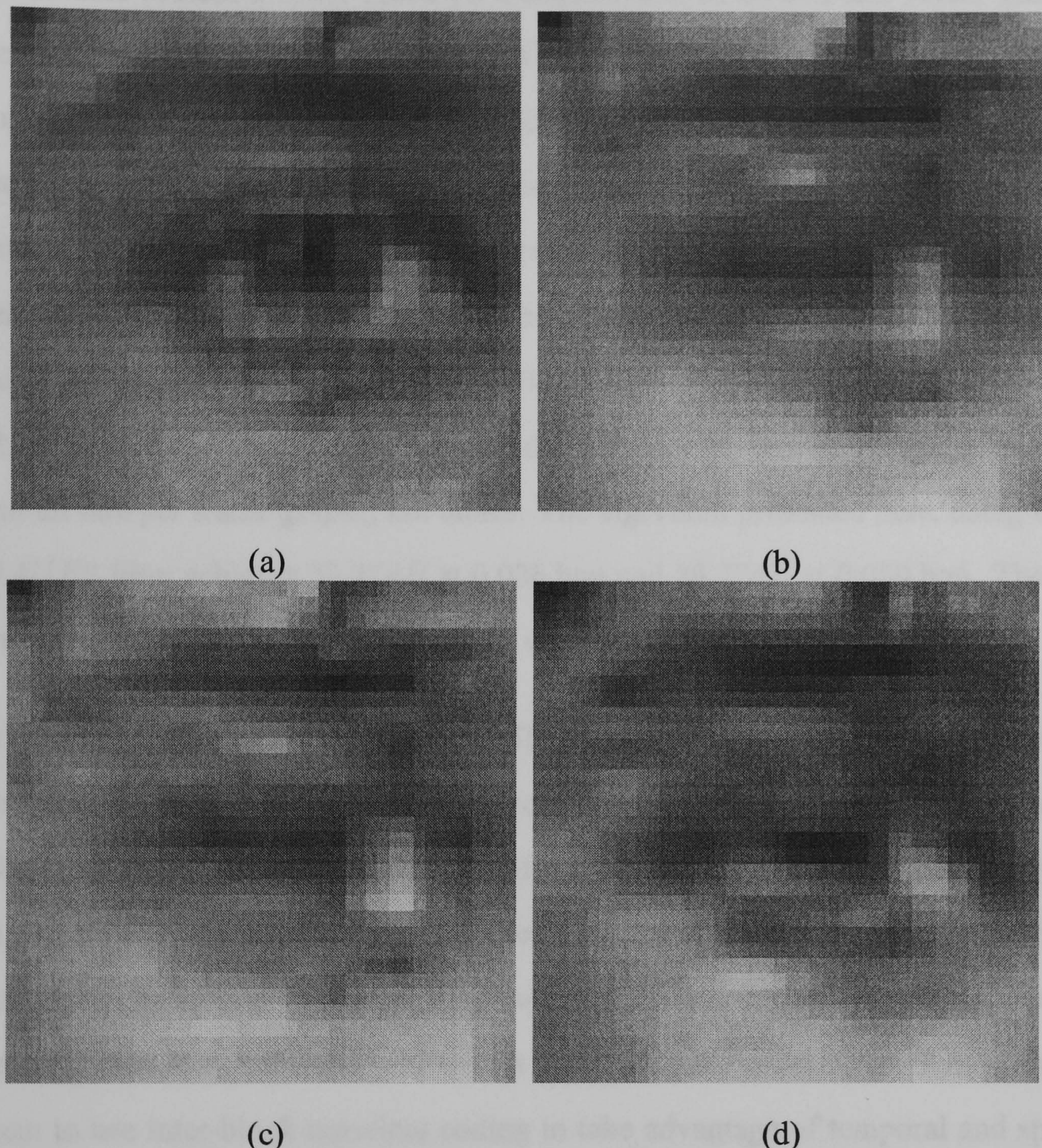


Figure 5.16: Detail from the eye area of frame 40 from reconstructions of the *Miss America* sequence. (a) Original (b) Linear quantizer coder, $DAUB4$ filter (c) Vector quantizer coder, $DAUB4$ filter (d) Vector quantizer coder, $DAUB8$ filter.

into bits per pixel, so that a simple comparison may be performed. In [CRA96], the authors present a coder based on a combination of DPCM and vector quantization methods. They produce results on QCIF (i.e. 144 $\times$ 276 pixel). Any comparison drawn from these results will not be perfect, since the resolution of the sequence plays an important role in determining the final quality of the reconstructed sequence. The authors produce results on the monochrome *Miss America* sequence at 10 frames per second. They achieve approximately $31.5dB$ at approximately 0.03 bpp and approximately $33.0dB$ at approximately 0.06 bits per pixel. These values are approximate because the authors only provide average PSNR and bit rate per frame graphs, not tables. The algorithm presented here, using the $DAUB8$ filter achieves $35.37dB$ at 0.036 bpp and $36.77dB$ at 0.059 bpp. These values represent gains of approximately $4dB$ in each case.

Podilchuk, Jayant and Farvardin [PJF95] presented a subband video coder based on a combination of tree structured vector quantization and ADPCM techniques. Their tree structured vector quantizer differs from the one presented in chapter 4 in that it only contains the data presented to it and so becomes unbalanced very quickly. The authors' node-splitting construction ensures that Euclidean close vectors share long common prefixes on their tree path codewords. This allows them to use inter-block noiseless coding to take advantage of temporal and spatial correlations. Any common prefixes were Huffman coded. Their results are presented in a very odd way; they simply state that

> ... the Miss America sequence looks very good encoded at 128kbps

(a) (b)

Figure 5.17: Reconstructed frames from *Miss America* sequence using Vector Quantization (DAUB16SYM @ 0.035bpp) (a) Frame 40, (b) Frame 41

...

The $128kbps$ figure is also misleading since they reserve $16kbps$ for audio data. The resulting $112kbps$ gives an average bit rate of $0.035bpp$ (for CIF format video at $30fps$). Subjectively speaking, the results from the coder presented here also look good, as demonstrated in figures 5.17(a) and 5.17(b). Note, however, that the coder presented in [PJF95] operates on sequences running at 30 frames per second. The *Miss America* sequence used in this work appears to be 15 frames per second. A higher frame rate increases temporal redundancy and so should produce better results.

In [Cho96], Chou presents a three dimensional subband coding scheme that uses a model of the human visual system. His model produces either *Just Noticable Distortion* or *Minimally Noticable Distortion* profiles from a model of the visual system. This allows the author's system to allocate bits to those coefficients which are more perceptually important. His temporal frequency decomposition uses the Haar wavelet and so only consists of two frames. The larger span of the temporal frequency decompostition presented here (which is the length of the underlying wavelet filter) provides a simple model of the sort of spatio-temporal frequencies that are important in the human visual system model. Indeed, Chou's results (which are, again, QCIF resolution at $15fps$) give approximately $33dB$ at $0.07bpp$ and $34dB$ at $0.10bpp$. The coder presented here, working on 256×256 sequences and using the $DAUB16SYM$ filter, will produce $37.06dB$ at $0.058bpp$ and $38.36dB$ at $0.097bpp$. Even taking into consideration the resolution difference, these results are still quite impressive.

5.7 Discussion

The results presented here are comparable with, and often better than, other coding schemes which are published at this time. Most current video coding schemes use motion compensation. Motion compensation schemes have problems, as detailed earlier in this chapter. For example, the motion vectors are often approximated to simple translations, and this often does not mimic reality. Furthermore, because motion compensation relies on prediction, it can be upset by changes in the sequence which do not adhere to the relatively simple model that the prediction imposes. A smooth video sequence will have many image elements that move only slightly between frames and the predictive nature of the motion compensation scheme will cope well with this. However, abruptly changing motions or flashing images do not fit the temporal prediction model and will not be handled well by a motion compensation scheme. The three dimensional wavelet transform, however, does not impose such a restrictive model on the data. The energy compaction and decorrelation properties of the transform mean that most of the information is concentrated in the lower frequency bands. Any rapid, non-smooth changes will produce data in the higher frequency bands. These high frequency components can be dealt with by the coder when necessary and the coder will degrade gracefully. The video coders presented here do suffer from some problems. The selection of the underlying wavelet filter affects the reconstructed image quality and bit rate. It appears that the longer symmetric filters give better results, both numerically and subjectively, probably due to their smoother nature. Longer filters also decorrelate the spatial data better, providing more scope for prediction and entropy coding. The simple linear quantizer would benefit from a more intelligent quantization scheme. Indeed, a Lloyd-Max quantizer or even several quantizers of differing granularities could be used for each spatio-temporal frequency band. This would allow the coder to take perceptual importance into account more precisely. The vector quantizer based coder could also benefit from a more robust VQ scheme. Normalising the vectors and sending a scale factor, as in the still image coder presented in chapter 3, would probably increase the quality of the reconstructed images. In summary, these results are very encouraging, but more work is needed to produce an optimised coder.

Chapter 6

Discussion, Conclusions and Future Work

The work in this thesis is concerned with the compression of still image and video compression using wavelets and the design of coders utilising the properties of the transform.

6.1 Conclusions

The analysis of the basis generated by iterated function schemes in chapter 2 is original. It demonstrates that, as defined by Jacquin [Jac90], fractal image compression cannot achieve perfect reconstruction of arbitrary signals. Furthermore, the dyadic averaging and downsampling preferred by fractal methods was shown to be sub-optimal. The optimal basis for fractal image compression was derived and it was shown that, up to a shift, the wavelet basis is the ideal basis for fractal image compression. Furthermore, it was shown that using an orthonormal wavelet transform would remove the need for iteration and, therefore, the constraint on the contractivity of the mappings.

The new paradigm for image coding introduced in chapter 3 builds directly on the analysis of chapter 2. The relation between fractal and wavelet methods is made concrete by the coder presented. The prediction down the wavelet tree is exactly as specified in the basis analysis. In order to compactly represent errors, the KL transform was used to create a codebook of error correction vectors which is complete on the space of errors. The preservation of local feature orientation was shown to be important by introducing a crude orientation estimate which is used to exclude vectors from the search for the minimal error prediction. The results produced by the coder are subjectively good at medium to high bit rates. There are no blocking artifacts, which are normally associated with block based image coders. Since the prediction occurs in the wavelet domain, any blocking artifacts will occur in the wavelet domain. When the inverse wavelet filter is applied, the quantization noise which created the blocking artifacts is 'spread out' over the length of the wavelet filter. This produces the higher frequency artifacts apparent at low bit rates. These artifacts, while still disturbing, are less preceptually important to the human visual system. The fractal-wavelet combination has thus been demonstrated to be a useful framework for predictive coding. Some of the problems associated with the system were addressed in the next chapter.

The conventional wavelet transform groups multiresolution features into similar orientations. The oriented wavelet transform presented in chapter 4 further refines this selectivity by producing 6 complex valued sub-bands. Even though it does not use the conventional quadrature mirror filter design, the transform retains many of its desirable properties. The tree structured vector quantizer presented in chapter 4 overcomes many of the limitations exhibited by earlier TSVQs. Indeed, it is fully on-line and there is no distinction between a *training* phase and a use phase. The inclusion of the distortion measure and representative vector at each node allows for variable rate coding or coding to a given error specification. The quantizer is relatively computationally simple, since the error at any node is calculated implicitly, rather than explicitly. The extra storage requirements of this method are far outweighed by the decrease in computation time. The results presented for the TSVQ, while limited, demonstrate its ability to match theoretical coding limits down to 1 bit per vector. The speed of the quantizer, coupled with the variable rate property make it ideal for image coding applications.

The results presented in chapter 4 are among the most promising in the image coding community. The PSNR distortion measure has often been shown to be a less than perfect measure of image quality. However, the coder presented here matches or exceeds the numerical results from all available research. It is impossible subjectively to compare image quality since the specifics of printing cannot be taken into account. Subjectively, the image reconstructions in chapter 4 are more pleasing than many other coding schemes, especially at very low bit rates.

The extension of predictive wavelet coding to video sequences is relatively straightforward. Chapter 5 presented a video coding scheme based around the three dimensional, Cartesian separable wavelet transform and the tree structured vector quantizer. An analysis of the spatio-temporal features present in each wavelet sub-band allows the codebook design to take into account both feature orientation and velocity. The final coding results again are among the best reported to date. The artifacts introduced at very low bit rates are disturbing, but not as disturbing as the heavy blocking artifacts introduced by, say, MPEG encoders.

6.2 Limitations and Further Work

All the work in this thesis in concerned with greyscale images. An obvious extension to each coding scheme would be to encode colour images and sequences. The extension should be relatively straightforward. The colour images could be represented in the YUV colour space. The coders presented here would operate, almost unchanged on the Y plane of these images. The U and V planes could be coded separately, to re-introduce colour.

For the still image coder, the simplistic nature of the quantization of the lower frequency wavelet coefficients is a major drawback. The most obvious alternative for encoding the low frequency coefficients is a vector quantization scheme. This would almost certainly only apply to the detail bands, leaving the lowpass band to be scalar quantized, as it is presently.

The effect that the underlying wavelet filter has on the reconstruction quality has been clearly demonstrated by the results presented in chapter 4. For example, the filter $BIORTH1810$ always performs significantly worse than the others. This could be attributed to the fact that this filter pair has a longer analysis filter than synthesis filter. The design of the wavelet filters to be used with the oriented wavelet transform has not been investigated in this work. The simple method used was quick and easy to implement, but does not take into account any psychovisual properties of the human visual system. The $DAUB8CONJ$ filter is based on Daubechies' 8 point filter [Dau88]. It consistently performs better, at least numerically, than the other filter pairs tested. This could be attributed to the properties of the underlying analysis filter. Much more work is required in this area.

The video coding algorithms presented here are not as well developed as the still image coding schemes. The argument about the coding of the lower frequency spatial coefficients above could be simply applied to the temporal lowpass in the three dimensional transform. In the version presented here, the vectors in the TSVQ are not normalised. Prediction vectors pulled from the tree structured vector quantizer are used 'as is'. A scheme akin to that of the still image coder, involving normalisation and scale factors, would probably increase the quality of the reconstructed sequences and certainly make rate control more intuitive. Furthermore, the video coding algorithm should better take into account the psychovisual properties of video. Simply discarding the highest temporal frequency band is certainly not optimal.

The methods described in this work combine many facets of established theories of image processing. It has been shown in this thesis that the combination is suitable for image compression and can match or beat many current coding schemes.

Appendix A

Required Proofs

A.1 The Contraction Mapping Theorem

The Contraction Mapping Theorem states for a contraction mapping T on a complete metric space X, there exists a *unique* fixed point $x^\star \in X$ such that $x^\star = T(x^\star)$. The unique fixed point is then

$$x^\star = \lim_{i\to\infty} T^i(x_0)\ , x_0 \in X$$

Proof

Given $x \in x$ and $n, m \in \mathcal{N}, n > m$ then,

$$d(f^m(x), f^n(x)) < d(f^{m-1}(x), f^{n-1}(x)) < s^m d(x, f^{n-m}(x)$$

Applying the Triangle Inequality repeatedly gives,

$$d(x, f^k(x)) \qquad \leq d(x, f^{k-1}(x)) + d(f^{k-1}(x), f^k(x)) \tag{A.1}$$

$$\leq d(x, f(x)) + d(f(x), f^2(x)) + \ldots + d(f^{k-1}(x), f^k(x)) \quad \text{(A.2)}$$

$$\leq (1 + s + s^2 + \ldots + s^k) d(x, f(x)) \quad \text{(A.3)}$$

$$\leq \tfrac{1}{1-s} d(x, f(x)) \quad \text{(A.4)}$$

Hence,

$$d(f^m(x), f^n(x)) \leq \frac{s^m}{1-s} d(x, f(x))$$

Now, take n and m to be sufficiently large to make $d(f^m(x), f^n(x))$ arbitrarily small. Then, $x, f(x), f^2(x), \ldots$ is a Cauchy sequence in X and X is complete. Therefore, the limit $x^\star = \lim_{n\to\infty} f^n(x) \in X$. f is, by definition, continuous and so

$$f(x^\star) = f(\lim_{n\to\infty} f^n(x)) = \lim_{n\to\infty} f^{n+1}(x) = x^\star.$$

$x^\star$ is unique since if $x_1^\star$ and $x_2^\star$ are both fixed points of f, then

$$d(f(x_1^\star), f(x_2^\star)) = d(x_1^\star, x_2^\star)$$

but

$$d(f(x_1^\star), f(x_2^\star)) \leq s \cdot d(x_1^\star, x_2^\star)$$

where $s < 1$, giving a contradiction. Hence $x^\star$ is unique.

A.2 The Collage Theorem

Let (X, d) be a complete metric space and let $B \in \mathcal{H}(X)$ and $\epsilon \geq 0$. Choose an IFS $\{X; w_0, \ldots w_N\}$ with contractivity factor $0 \leq s < 1$ such that

$$h\left(B, \bigcup_{n=0}^{N} w_n(B)\right) \leq \epsilon,$$

where $h(d)$ is the Hausdorff metric. Then,

$$h(B, A) \leq \frac{\epsilon}{1-s}.$$

Proof

The proof is a direct corollary of equation A.4 by allowing $k \to \infty$.

Predictive Wavelet Transform Coding : Unifying Fractal and Transform coding

Ian Levy, Roland Wilson
Department of Computer Science, University of Warwick, Coventry, UK

ABSTRACT: *It is shown that fractal image coding can be placed in a conventional predictive framework by using a wavelet representation. This has the significant advantages of simplifying decoding by avoiding iteration and of relating the technique to traditional coding methods. The resulting coder is a combination of transform and predictive techniques. Results are presented for a range of bit rates and images, which show that the new method has some potential as a data compression technique for still images.*

1 INTRODUCTION

In recent years, both fractal coding and wavelet transform coding algorithms have been the subject of extensive research [1] [6] [3] [7] [9]. Although they share a reliance on the scaling properties of the edge features which dominate normal images, there has previously been little attempt to marry them (see [8] however).

In a wavelet decomposition, the image is recursively decomposed into frequency bands. The wavelet subbands contain the high frequency components of the image in the horizontal, vertical and diagonal directions. These components can easily be vector quantized with low visible distortion because (it is argued) the wavelets are better matched to edge features than the sinusoidal basis functions used in earlier transform coders. Fractal image coding relies on the concepts of self-similarity and affine maps, which may be used to describe how to transform one image region into the other. The image is tiled with such regions and the collection of the affine maps which results is known as the *fractal code* for the image. An approximation to the original image is reconstructed from the fractal code by iterating the affine maps on any initial image [6] [3] [2].

Recently, there have been several attempts to combine fractal and transform compression methods [8] [4] [5]. These have, for the most part, used DCT basis vectors to represent the error resulting from the fractal approximation. The purpose of this paper is to describe a system combining wavelet and fractal coding.

2 COMBINING THE WT AND FRACTAL COMPRESSION

The new method can be seen as a predictive coder, in which blocks of data representing larger scales are used to predict those at smaller scales. Errors are coded using a KLT coder operating in the wavelet domain. The novel algorithm presented combines parts of both wavelet and fractal coding. The image to be coded is decomposed by the wavelet transform to a given level, say l, known as the *base* level. The first level to be approximated, called the *domain level*, is then decided. This is usually the level below the base level. Then, the *range level* is the level from which the domain level shall be approximated (ie the level above the domain level). Levels from l to the range level are quantized using a linear quantizer for the highpass coefficients and a separate quantizer for the lowpass coefficients at the highest level. The quantized approximation is used as the range level of coefficients to reduce errors at the decoder. Each of the bands in the domain level is partitioned into non-overlapping domain blocks. Let $\{D_i^{HL}\}$, $\{D_i^{LH}\}$ and $\{D_i^{HH}\}$ be the domain blocks from each subband. For each domain block set $\{D_i^{HL}, D_i^{LH}, D_i^{HH}\}$, a corresponding range block set $\{R_i^{HL}, R_i^{LH}, R_i^{HH}\}$ must be found which minimizes the error $\epsilon = ||D_i^{HL} - \hat{D}_i^{HL}||^2 + ||D_i^{LH} - \hat{D}_i^{LH}||^2 + ||D_i^{HH} - \hat{D}_i^{HH}||^2$ where $\hat{D}_i^X = \iota_i(\alpha_i \cdot \hat{R}_i^X)$, $\hat{R}_i$ is a normalised range block, $\alpha_i = \frac{<\hat{R}_i^{HH}, D_i^{HH}> + <\hat{R}_i^{HL}, D_i^{HL}> + <\hat{R}_i^{LH}, D_i^{LH}>}{<\hat{R}_i^{HH}, \hat{R}_i^{HH}> + <\hat{R}_i^{HL}, \hat{R}_i^{HL}> + <\hat{R}_i^{LH}, \hat{R}_i^{LH}>}$ and $\{\iota_i\}$ is the dihedral group of the square. Hence, the blocks from the three subbands are scaled by the same factor.

Since an orthogonal wavelet transform is used, the subbands are all orthogonal. Hence, no iteration of the affine maps is required. Since there will be no iteration of the maps in the decoding process, the value of α_i is unconstrained. Given a domain block size d and and search radius r, we define the *search region* as the subset of the subband $\mathcal{I}$ given by $([x-rd, x+rd] \times [y-rd, y+rd]) \cap \mathcal{I}$. A spiral search from the center of the search region is used to address the range blocks. This method constrains the maps so that each range block R_i^X

($X \in \{HH, HL, LH\}$) is in the same realtive position, but in different subbands. The reasoning behind this is that if two blocks are similar, then their horizontal, vertical and diagonal components must also be similar.

This process is then repeated on successive levels of the wavelet decomposition. The approximation of the domain level generated by the coder thus becomes the range coefficients for the next iteration. Separate affine map coefficients are generated between levels, up to level 1. As it stands, the coder attempts to extrapolate a block in the range level to a block in the domain level in each subband. However, if there is not a good match between wavelet levels, the overall reconstruction error will increase for each bad block. To overcome this, block projection is performed after the fractal coding on each level to reduce errors, in the same vein as Huang and Gharavi-Alkhansari [5]. The present method, however, is simpler, since it is based on the Karhunen-Loève transform for the symmetrised error vectors. A block diagram of the coder is shown in figure 1. It is worth noting that the coder can operate in a variety of modes, depending on application. It is possible to vary the block size, enable or disable the Karhunen-Loève transform sub-system and enable or disable the block searching (ie restrict the search region to a single block). Obviously, different configurations will be suitable for different applications.

The parameter sets that describe the blockwise maps and the basis projection coefficients are linearly quantized and encoded using an order zero arithmetic coder. A *rate value* determines how many bins the quantizer uses and how many symbols the arithmetic coder can encode. Reconstruction proceeds much the same as for standard fractal coding. The major difference is that this method does not require iteration of the maps. If the Karhunen-Loève transform sub-system is enabled, then the projected error vectors are added to the reconstructed approximation of the block. An inverse wavelet transform from the base level reconstructs the original image.

3 RESULTS

In this section, sample compressed and reconstructed images at low bit rates with and without searching on the standard *Lena* and *Boats* images are presented. The test images are 512×512 pixels in 8-bit greyscale. For the tests, a block size of 4 pixels and the Daubechies 8 point filter are used and the Karhunen-Loève transform sub-system is enabled. The results of these tests are shown in the Table 1 and the following figures.

For comparison, Figure 6 shows the Lena image coded at 0.22 bits per pixel with the JPEG system. This image has a PSNR of 30.71 dB and much more visible artifacts. Shapiro's EZW coder [9] codes Lena at 0.25 bpp with a Peak-RMS SNR of 33.17 dB and at 0.125 bpp with a PSNR of 30.23 dB. The coder presented here, achieves 31.52 dB in 0.20 bpp, a comparable figure.

As can be seen from the above results, at low bit rates, the search does not improve the coder performance. However, further tests have shown that at higher bit rates, a search is advantageous. Using a full search, the Lena image can be coded at 1.49 bpp with a PSNR of 38.67 dB. With no search, the image is coded in 1.44 bpp with a PSNR of 38.07 dB. At these bit rates, a gain of 0.6 dB for 0.05 bpp is significant.

4 DISCUSSION

This paper has shown that it is possible to place fractal coding into a multiresolution predictive framework by using an orthonormal wavelet transform. Initial results from the coder show that the method has real promise in applications. One significant problem with this technique is the excessive search time. For the highest quality (ie a full search), coding time is prohibitive. However, possible avenues of further research include tree structured searches, Self Organising Feature Maps and possibly other neural network approaches. Another area requiring more work is the choice of wavelets - the lack of symmetry of the separable Daubechies wavelets clearly has an impact on the quality of the results.

References

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using Wavelet Transform. In *IEEE Transactions on Image Processing*, volume 1, 1992.

[2] M. Barnsley and S. Demko. Iterated Function Systems and the Global Construction of Fractals. In *Proc. Royal Society of London*, 1985.

[3] M. Barnsley and L. Hurd. *Fractal Image Compression.* AK Peters, 1992.

[4] K. Barthel and T. Voyé. Adaptive Fractal Image Coding in the Frequency Domain. In *Proceedings of International Workshop on Image Processing*, 1994.

[5] M. Gharavi-Alkhansari and T. Huang. Fractal-Based Techniques for

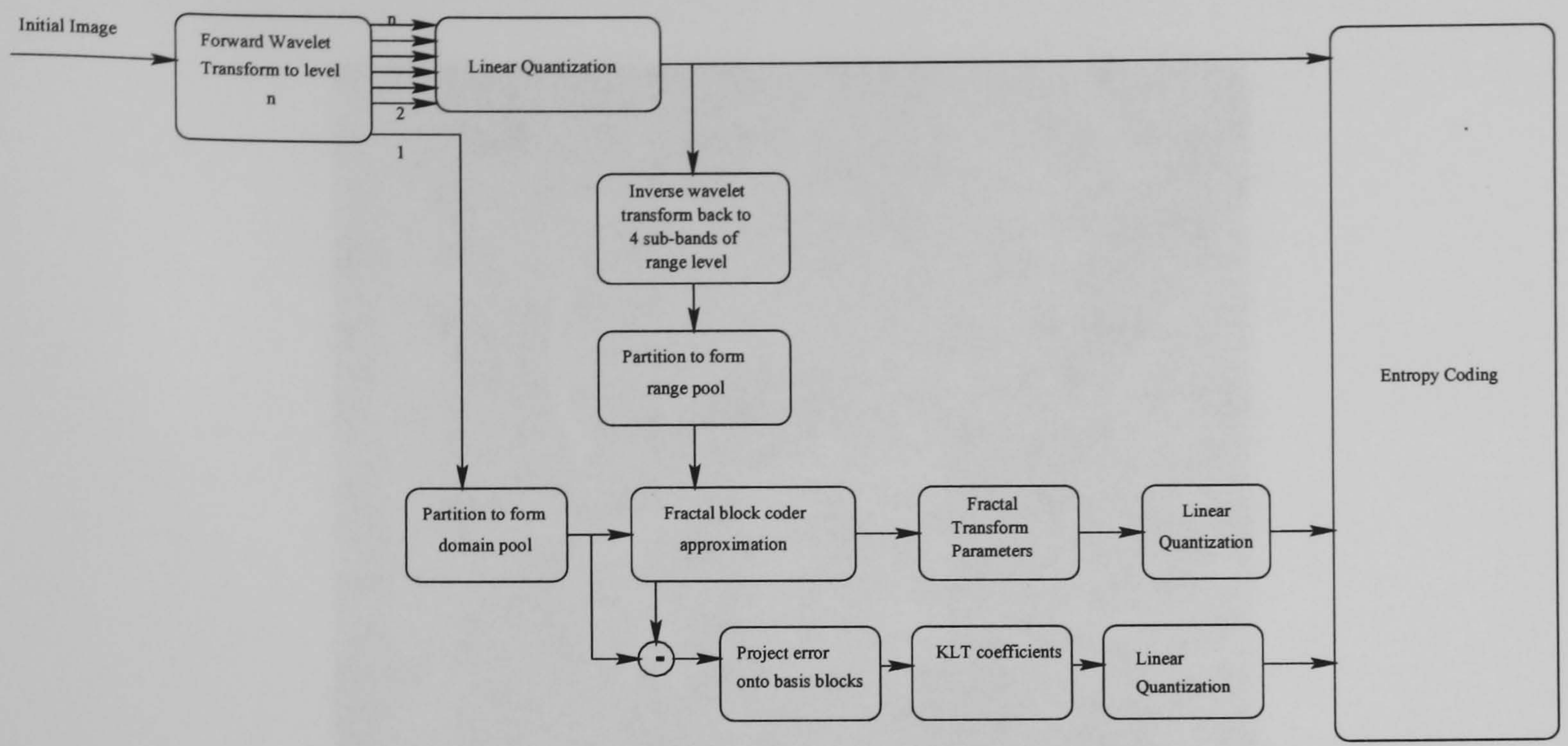


Figure 1: Coder block diagram

Image	Search	Rate (bpp)	Compression Ratio	Peak-RMS SNR
Lena	Full	0.25	32:1	31.61 dB
Lena	None	0.20	40:1	31.52 dB
Boats	Full	0.34	23.5:1	29.95 dB
Boats	None	0.28	28.5:1	29.94 dB

Table 1: Coder test results

Figure 2: Reconstruction of Lena with no search

Figure 3: Reconstruction of Lena with full search

Figure 4: Reconstruction of Boats with no search

Figure 5: Reconstruction of Boats with full search

Figure 6: JPEG Reconstruction at 0.22 bpp, PSNR 30.71

a Generalized Image Coding Method. In *Proc. IEEE ICASSP*, 1994.

[6] A. E. Jacquin. A Novel Fractal Block-Coding Technique for Digital Images. In *Proc. ICASSP '90*, 1990.

[7] D. Monro and F. Dudbridge. Fractal Approximation of image Blocks. In *Proc. IEEE ICASSP*, 1992.

[8] R. Rinaldo and G. Calvagno. Image Coding by Block Prediction of Multiresolution Subimages. Technical report, Universitá di Padova, 1995.

[9] J. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. In *IEEE Transactions on Signal Processing*, volume 41, 1993.

Symmetry and Wavelet Transforms for Image Data Compression

R Wilson, I Levy and P R Meulemans
Department of Computer Science,
University of Warwick,
Coventry CV4 7AL.

Abstract

This paper describes work in the application of wavelet transforms to the data compression of both still images and image sequences. The main principle underlying the work is the representation of the natural symmetries of image data, which form a subgroup of the 2-d affine group, itself an approximation in 2-d of the motions resulting from perspective projection of the rigid motions in 3-d. It will be shown how the use of a suitable wavelet transform can simplify the representation of these motions, to good effect in two data compression applications. The first is a recasting of fractal compression into a more conventional predictive framework, using an orthonormal wavelet basis. The second makes use of an overcomplete wavelet transform in estimating motions for video coding.

1 Symmetry and Image Representation

The use of wavelet transforms (WT) in image analysis and data compression is one of the more significant developments in the subject over the last ten years or so. The introduction of orthonormal wavelet bases by Daubechies [1] has led to a wide range of techniques using various types of wavelet representation [2, 3, 4, 11]. One obvious feature of wavelets, which distinguishes them from other image representations, is their symmetry properties, the continuous WT in $1-d$ representing the coherent states of the affine group, for example [1]. Such symmetries as rotation, translation and dilation, which constitute a subgroup of the $2-d$ affine group, are clearly important in image analysis and ought to be considered, when selecting an image representation. Indeed, the widespread use of Fourier transforms in signal analysis is directly attributable to their role in the representation of the translation group [8]. Symmetries are reflected in statistical measures such as the autocorrelation tensor, revealing themselves in the eigen-analysis of that tensor, ie. the Karhunen-Loève transform (KLT) of the signal. Over the years, much attention has been paid in image data compression to the statistical aspects of the problem (see eg. [7]), but relatively little to symmetry. A notable exception to this can be found in the work of Jacquin on iterated-function data compression [5]. His work draws heavily on the ideas of self-similarity under the affine group, which were expounded by Barnsley [6] and although in its simple form it has its weaknesses, it nevertheless represents a direct way of incorporating symmetries into an image description. More recently, some work on texture description has taken the idea of self-similarity in a different direction, building on a $2-d$ affine motion model to generalise the simple idea of a texture as a periodic placement of a texture element [10]. In the compression of video sequences, of course, it has long been recognised that motion estimation is a central component [9], but limited computational power has meant that much of the work has focused on simple methods such as block matching to provide the velocity estimates. Inevitably this leads to poor estimates, since the block structure, though it underlies the current data compression algorithms such as JPEG and MPEG [9], is hardly symmetry-adapted. While use of multiresolution processing can mitigate these effects, it is no substitute for a properly designed image representation.

The work presented in this paper is concerned with exploiting the symmetry properties of certain types of wavelet transform, to see how these might be used in applications such as image data compression. After a brief discussion of the underlying principles, two examples will be used to illustrate the importance of symmetry in image representation. The first is based on an orthonormal wavelet transform, which will be shown to lead to a simpler and more elegant approach to fractal image coding, with results approaching those of the best systems currently reported for still image compression. The second uses an *overcomplete* WT, the multiresolution Fourier transform (MFT), in video coding using a combination of featureless and feature-based motion estimation. The key property of the MFT in this application is that it reflects the structure of the $2-d$ affine group, simplifying estimation of the motion parameters. Results of motion estimation using this technique will be used to illustrate its potential. The paper is concluded with a discussion of the relationship between symmetry and statistics in image representation.

2 Representing Symmetry in Images

The role of symmetry in image representation is widely misunderstood to be limited to axial or rotational symmetries, such as occur in simple geometric shapes. In the present context, a broader definition is needed - one which reflects the role of $2-d$ or $3-d$ motions. To this end, suppose there are two images, or rather two image patches, labelled $f_1(\vec{x}), f_2(\vec{x})$ respectively, where the image co-ordinates are $\vec{x} \in R^2$. They might be parts of the same image or from two images in a temporal sequence, for example. To say that they are related by a symmetry is to assert that there exists a co-ordinate transformation $\boldsymbol{T}$ such that

$$f_2(\vec{x}) = f_1(\boldsymbol{T}^{-1}\vec{x}) \tag{2.1}$$

where the exponent '-1' is used to indicate the inverse, which will be assumed to exist. (2.1) asserts that the two patches are related by a *motion*: the second can be found by simply moving the first appropriately. For example, if $f_1(\vec{x})$ represents a horizontal edge and $f_2(\vec{x})$ a vertical one, then the transformation is just a rotation by $\pi/2$. While ideal image features like edges may exhibit such perfect symmetry, a more realistic model includes an element of approximation, replacing (2.1) by

$$f_2(\vec{x}) = f_1(\boldsymbol{T}^{-1}\vec{x}) + \nu(\vec{x}) \tag{2.2}$$

where the *residual*, $\nu(\vec{x})$, expresses the approximation error. This formulation can also be viewed as a rather general model for image data compression: the patch f_2 is *predicted* from f_1 and the prediction error is the residual. Typically, most of the work in compression algorithms, both intraframe and interframe, is done by the residuals - the predictor structure is fixed or has relatively few degrees of freedom. Notable exceptions to this are the iterated function system (IFS) for still image compression [5] or motion compensation schemes for video compression, both of which typically spend more information on the transformation than on the residuals. This raises the propsect of choosing the transformation, among some admissible set of motions, to minimise the average residual error energy:

$$\boldsymbol{T}_{opt} = \arg\min_{\boldsymbol{T}} \sum_{\vec{x}} (f_2(\vec{x}) - f_1(\boldsymbol{T}^{-1}\vec{x}))^2 \tag{2.3}$$

where the sum is over all pixels within the patch. The main problem with this formulation is the implied search of the parameter space: the full affine group is a 6-parameter continuous group; even the discrete subgroups used below constitute a huge space. For example, in a $N \times N$ pixel patch, there are of the order of N^2 translations. Identifying the minimum error fit using cross-correlation requires N^2 multiplications per position, giving an overall burden of N^4. But if the computation is done in the Fourier domain, (2.3) can be replaced by

$$\vec{\tau}_{opt} = \arg\max_{\vec{\tau}} \int d\vec{\omega} \hat{f}_2(\vec{\omega}) \bar{\hat{f}}_1(\vec{\omega}) e^{j\vec{\omega}.\vec{\tau}} \tag{2.4}$$

where $\hat{f}$ is the FT of f and $\bar{\cdot}$ denotes complex conjugate. This trick reduces the computation to one of order N^2. This results from the diagonalisation of translations by the FT. It is this sort of computational issue which makes the choice of image representation significant.

When translations are the only motion, their representation by cyclic shifts is a convenient simplification, which leads to use of discrete FT methods based on (2.4), but the general affine case is less obliging - the group is not abelian and no representation will provide a full diagonalisation. Moreover, it is seldom the case that motions of the whole image are involved - most relevant motions apply *locally*, rendering simple Fourier methods ineffective. The best which can be done in general is to require that the function set $\mathcal{W} = \{w_{a,b}(\vec{x})\}$ is *invariant* to the group operations, ie.

$$\boldsymbol{T}\mathcal{W} = \{w_{a,b}(\boldsymbol{T}^{-1}\vec{x})\} = \mathcal{W}, \quad \boldsymbol{T} \in \mathcal{T} \tag{2.5}$$

In other words, the vectors $w_{a,b}(\vec{x})$ are *coherent states* of the group [1]. The significant feature of this invariance is that any motion can be represented by a motion of the parameters

$$w_{a,b}(\boldsymbol{T}^{-1}\vec{x}) = w_{T(a),T(b)}(\vec{x}), \quad \boldsymbol{T} \in \mathcal{T} \tag{2.6}$$

where $T(a)$ is a mapping of the parameter a. Among other things, this implies that the motions and the parameter space have the same cardinality, whether finite or infinite. If such a representation exists, then the problem of identifying motions avoids the computation of $f(\boldsymbol{T}^{-1}\vec{x})$ for each transformation. As an example, the continuous $1-D$ wavelet transform of a signal is [1]

$$Wf(x,\sigma) = \frac{1}{\sqrt{\sigma}} \int_y dy\, f(y) w(\frac{y-x}{\sigma}) \tag{2.7}$$

where $w(.)$ is the so called 'mother wavelet'. This two-parameter transform is the natural representation for the $1-d$ affine group, in the sense of (2.5), since if $(\alpha,\beta) : x \mapsto \beta x + \alpha$ is the general group element, then its action on the WT is just

$$Wf(x,\sigma) \overset{(\alpha,\beta)}{\mapsto} \beta W f(\frac{x-\alpha}{\beta}, \frac{\sigma}{\beta}) \tag{2.8}$$

which is no more than a 'relabelling' of the transform. The commonest form of $2-d$ WT is the Cartesian product of $1-d$ transforms in the horizontal and vertical directions

$$Wf(\vec{x},\sigma) = \frac{1}{\sigma} \int d\vec{y}\, f(\vec{y}) w(\frac{\vec{y}-\vec{x}}{\sigma}) \tag{2.9}$$

where the wavelet $w(.)$ is separable. This has rotations by multiples of $\pi/2$ and axial symmetry about the vertical axis, in addition to translation and uniform dilations, as its natural group. Continuous WT's adapted to the full rotation group, translations and dilations have also been described [12].

In the general affine case, a suitable choice of representation is the over-complete wavelet transform called the multiresolution Fourier transform (MFT) [11],which gives estimates of the Fourier spectrum of image patches at a range of scales. The MFT for continuous $2-d$ signals is defined as

$$Mf(\vec{x},\vec{\omega},\boldsymbol{L}) = |\boldsymbol{L}|^{\frac{1}{2}} \int d\vec{\xi}\, f(\vec{\xi}) w(\boldsymbol{L}^{-1}(\vec{\xi}-\vec{x})) e^{-j\vec{\omega}.\vec{\xi}} \tag{2.10}$$

where $w(.)$ is a window function, $\boldsymbol{L}$ is an invertible linear transformation, $\vec{x}$ is the spatial and $\vec{\omega}$ the frequency co-ordinate. This represents a generalisation of the form given in [11], to accommodate the affine symmetry model. As a signal representation, the MFT is not square-integrable, but it has a form of inverse given by

$$f(\vec{x}) = \frac{1}{4\pi^2} \int d\vec{\omega} d\vec{\xi} d\boldsymbol{L}\, |\boldsymbol{L}|^{\frac{1}{2}} \rho(\boldsymbol{L})\, Mf(\vec{\xi},\vec{\omega},\boldsymbol{L}) w(\boldsymbol{L}^{-1}(\vec{\xi}-\vec{x})) e^{j\vec{\omega}.\vec{x}} \tag{2.11}$$

where $\rho(\boldsymbol{L})$ is a *density* over the linear group

$$\int d\boldsymbol{L}\ \rho(\boldsymbol{L}) = 1 \tag{2.12}$$

and provided $w(.)$ satisfies the weak requirement

$$\int d\vec{x}\ w(\vec{x})w(\vec{x}) = 1 \tag{2.13}$$

This is readily proved by an extension of the arguments in [11]. In the present context, the MFT has two important properties: symmetry and locality.

First, consider the action of an affine group element $(\boldsymbol{A}, \vec{a})$ on the MFT. From (2.10), if $(\boldsymbol{A}, \vec{a}) : \vec{x} \mapsto \boldsymbol{A}\vec{x} + \vec{a}$, then its action on the MFT is

$$Mf(\vec{x}, \vec{\omega}, \boldsymbol{L}) \overset{(\boldsymbol{A},\vec{a})}{\mapsto} |\boldsymbol{A}|\ e^{-\jmath\vec{\omega}.\vec{a}}\ Mf(\boldsymbol{A}^{-1}(\vec{x} - \vec{a}), \boldsymbol{A}^T\vec{\omega}, \boldsymbol{A}^{-1}\boldsymbol{L}), \tag{2.14}$$

which is an obvious generalisation of (2.8). More significantly, for any finite energy image $f(\vec{x})$ and group element $(\boldsymbol{A}, \vec{a})$, there is *some* window scale, σ, for which

$$Mf(\vec{x}, \vec{\omega}, \boldsymbol{L}) \overset{(\boldsymbol{A},\vec{a})}{\mapsto} |\boldsymbol{A}|\ e^{-\jmath\vec{\omega}.\vec{a}}\ Mf(\vec{x}, \boldsymbol{A}^T\vec{\omega}, \boldsymbol{A}^{-1}\boldsymbol{L}) + O(\frac{1}{|\boldsymbol{L}|}), \quad |\boldsymbol{L}| > \sigma \tag{2.15}$$

In other words, the translation can be determined from the *phase* and the linear transformation from the magnitude of the MFT's of the original and transformed images. This observation underpins the use of the MFT in video coding.

Locality is a key issue in signal modelling, which goes beyond the simplistic argument that because WT's use basis functions of arbitrary size, they avoid the limitations imposed by the uncertainty principle. It is exemplified in (2.15) by the $O(\frac{1}{\sigma})$ term on the right hand side: *any* model of images is valid only over some range of scales. In the above case, the window scale has to be large enough to allow the factoring of the group action into its two components - translation and linear transform. But there is a catch: the affine model itself is an approximation, only valid over a range of scales up to some largest scale; beyond that, multiple motions will be encountered, due for example to the effects of perspective, occlusions and so on. Since the range of usable scales varies across images and even within many images, it requires a representation allowing such variation of scale *without* sacrificing invertibility.

In this respect, the MFT has a significant advantage over the simpler forms of the WT: the window scale can be selected to get the best model fit; it is not tied to the basis functions. This is why the inversion formula contains a density over $\boldsymbol{L}$ - the lack of a conventional inverse, far from being unfortunate side-effect of the definition, is essential in allowing the selection of the optimum window for a given data set. To be more specific, consider again the model of (2.1), expressed in terms of the corresponding MFT's and using (2.15)

$$Mf_2(\vec{x}, \vec{\omega}, \sigma\boldsymbol{I}) = |\boldsymbol{A}|\ e^{-\jmath\vec{\omega}.\vec{a}}\ Mf_1(\vec{x}, \boldsymbol{A}^T\vec{\omega}, \sigma\boldsymbol{A}^{-1}) + \nu(\vec{x}, \vec{\omega}, \sigma\boldsymbol{I}) \tag{2.16}$$

where the patch being predicted $f_2(\vec{x})$ is circular, reducing the general case to the scale σ only. Now choose the scale to maximise the likelihood, ie.

$$\sigma_{opt} = \arg\max_s \left(\max_T prob(f_2(\vec{x}), ||\vec{x}|| < r | f_1(\boldsymbol{T}^{-1}\vec{y}), ||\vec{y}|| < s \leq r) \right), \qquad (2.17)$$

where $prob(.|.)$ is the conditional density. A simple model illustrating the idea is that the two signals are related via (2.16), for scales $s \leq \sigma$. Following the standard approach to the problem [14], with appropriate normalisation, for jointly normal, white $f_1(.), f_2(.)$, the log-likelihood is maximised by choosing the scale and affine parameters according to

$$\begin{aligned} \sigma_{opt} \quad = \quad & \arg\max_s \left(\max_{(\boldsymbol{A},\vec{a})} 2|\boldsymbol{A}| \int d\vec{\omega} M f_2(\vec{x}, \vec{\omega}, s\boldsymbol{I}) \bar{M} f_1(\vec{x}, \boldsymbol{A}^T\vec{\omega}, s\boldsymbol{A}^{-1})\, e^{j\vec{\omega}.\vec{a}} - \right. \\ & \left. |\boldsymbol{A}|^2 \int d\vec{\omega} |M f_1(\vec{x}, \boldsymbol{A}^T\vec{\omega}, s\boldsymbol{A}^{-1})|^2 \right), \end{aligned} \qquad (2.18)$$

which takes advantage of (2.16) to factor out the translations, which are estimated using cross-correlation, once the linear transformation has been identified. By using the MFT, the scale can be handled as an unknown parameter - the best scale being selected in a data-dependent fashion, without losing invertibility. The combination of symmetry and locality marks the MFT as uniquely fitted to motion estimation.

In practice, signals are sampled, with dilations being restricted to the subgroup $s_i = 2^i$ and the affine group actions are approximated by a linear action on the signal space, so that (2.2) is replaced by the vector form

$$\boldsymbol{f}_2 = \boldsymbol{T}\boldsymbol{f}_1 + \boldsymbol{\nu} \qquad (2.19)$$

where $\boldsymbol{T} : R^{N^2} \mapsto R^{N^2}$ is a linear operator. The computational complexity of this operator depends on the structure of the matrix $\boldsymbol{T}$, which again is greatly simplified by the choice of image representation. It is perhaps worth noting that (2.19) is formally identical to the standard predictive models used so widely in image processing [7]. The significant difference from the use here is that in the conventional predictive models the transformations are effectively symmetrised by averaging over the group, leading to a predictor which is lowpass and, by definition, carries no information - it is spatially invariant.

3 Applications

As a first example, the use of affine maps forms the basis of fractal coding, in which 4×4 blocks of pixels, called *range* blocks are represented as 'motions' of 8×8 *domain* blocks. The allowable transformations are products of inversion about the $y-$axis, $\boldsymbol{F}$, rotations $\boldsymbol{R}$ by multiples of $\pi/2$ and scaling by a factor of 2 to reduce the block size, which is accomplished by a 2×2 block averaging operator $\boldsymbol{A} : R^{64} \mapsto R^{16}$. To these co-ordinate transformations are added magnitude scaling by $s < 1$ and translation by a constant c

$$\boldsymbol{f}_r = s_{dr} \boldsymbol{R}^i \boldsymbol{F}^j \boldsymbol{A} \boldsymbol{f}_d + c_{dr} \mathbf{1}, \quad 0 \leq i < 4,\ 0 \leq j \leq 1 \tag{3.1}$$

where $\boldsymbol{f}_r$ is a 4×4 range block and $\boldsymbol{f}_d$ an 8×8 domain block. With s, c quantized appropriately, most of the computation and the bit rate are consumed by locating the best domain block for a given range block. The constraint on s guarantees that the mapping thus defined on the image is a contraction, whose fixed point is an approximation to the original image [5]. This is essential because the image is reconstructed from an arbitrary initial image by iteration of the image-image mapping defined by the domain-range block transformations:

$$\boldsymbol{f}_n = \boldsymbol{L} \boldsymbol{f}_{n-1} + \boldsymbol{e} \tag{3.2}$$

where $\boldsymbol{L}$ is the linear transformation and $\boldsymbol{e}$ the 'error' comprising the piecewise constant image represented by the last term in (3.1). Note that both $\boldsymbol{L}$ and $\boldsymbol{e}$ are independent of n in this case. In the limit, because of the fixed-point theorem,

$$\boldsymbol{f}_\infty = \lim_{n \to \infty} \boldsymbol{T}^n \boldsymbol{f}_0 \tag{3.3}$$

In practice, convergence is generally obtained in a few tens of iterations, but it is by no means simple to achieve a specified rate-distortion trade-off with the method.

If an orthonormal wavelet transform is used, however, (3.1) is replaced by

$$\boldsymbol{f}_{n,r} = s_{dr} \boldsymbol{R}^i \boldsymbol{F}^j \boldsymbol{f}_{n-1,d} + e_{n,r} \tag{3.4}$$

where $\boldsymbol{R}, \boldsymbol{F}$ are as in (3.1) and the scale factor s_{dr} is no longer constrained to the range $|s| < 1$. Because the WT is adapted to dilations (cf. (2.8)), both range and domain blocks are 4×4 pixels in size; if an orthonormal WT is used, iteration is redundant: errors at level n are orthogonal to the data on level $n-1$ from which the prediction was derived. Consequently, the residuals are coded using the optimal orthonormal basis - the KLT. This reduces the iterated function coder to a form of predictive vector quantisation (VQ) across scales and avoiding the most visible artefacts associated with the original IFS coder, which were due to the block structure it used. In the experiments performed with this coder, 512×512 pixel images have been coded using a Daubechies 16-pt wavelet pair. Levels above the second scale were coded using scalar quantization; 4×4 quantized blocks on level 3 were used to form the domain pool, from which 256 vectors were selected for tree-structured VQ applied to levels 2 and 1. All three wavelet bands at each scale were coded together, so that the vectors in the VQ were $48-d$. Figure 1 shows the coder's performance on the widely used 'Lena' image, coded at 0.16 bits per pixel (bpp) and 0.25bpp. The results compare favourably with those reported in [3] and other WT-based coders and are a significant improvement on the standard JPEG algorithm, both in terms of signal-noise ratio and visual quality.

A second application for symmetries in coding is in the compression of image sequences: much of what appears in frame i in a sequence results from movement of what appeared in frame $i-1$. The relevant symmetry group in this case is the full $2-d$ affine group, which is a linearisation of the projective group

$$f_i(\vec{x}) = f_{i-1}(\boldsymbol{A}^{-1}(\vec{x}-\vec{a})) + \nu_i(\vec{x}). \tag{3.5}$$

The algorithm uses both featureless and feature-based estimation: the featureless component is based on a discrete form of (2.18), in which a coarse-fine search is terminated when the likelihood for the affine model exceeds a threshold; this is corrected by a feature-based local update, designed to reduce errors at occlusion boundaries. The featureless estimates are sufficiently accurate to avoid the usual problem with feature-based approaches - the *correspondence* problem, in other words, identifying which feature in frame $i-1$ corresponds to which in frame i.

Identification of the motion parameters uses the discrete MFT. The use of a form of FT allows the factoring of the estimation into two parts cf. (2.16): the translation is estimated by cross-correlation

$$\hat{\vec{a}} = \arg\max_{\vec{y}} \sum_{\vec{x}} f_i(\vec{x}) f_{i-1}(\boldsymbol{A}^{-1}(\vec{x}-\vec{y})), \tag{3.6}$$

after the linear transform has been found by identifying from the Fourier transformed patches , $Mf_i(\vec{x},\vec{\omega},\sigma)$, a pair of axes based on 2-Means clustering of the spectral magnitudes

$$\{\vec{\chi}_{i,j},\ j=1,2\} = \arg\min_{\vec{\omega}_1,\vec{\omega}_2} \sum_{\vec{\omega}} |Mf_i(\vec{x},\vec{\omega},\sigma)|^2 \min_k \|\vec{\omega}-\vec{\omega}_k\|^2 \tag{3.7}$$

The matrix $\boldsymbol{A}$ is found from the co-ordinate transform between the two

$$\vec{\chi}_{2,j} = \boldsymbol{A}^T \vec{\chi}_{1,j},\quad j=1,2 \tag{3.8}$$

These area affine estimates are illustrated in Figure 2(b), for a pair of frames from the 'Table Tennis' sequence. Note how the window scale is smaller near occlusion boundaries around the bat and ball, for example. Updating of the featureless estimates uses the linear features estimated and tracked from frame to frame using a modification of the procedures described in [11]. First, any feature in frame $i-1$ (Fig. 2(a)) is moved using the appropriate affine estimate, then a local search is performed to correct the motion estimate. Because the features are approximately linear, only the rotation and translation perpendicular to the feature are updated, but this prevents the accumulation of errors at occlusion boundaries, as the updated features in Fig. 2(c) show. The last stage in the process is to amend the running feature map by deleting any features which have vanished and adding any new ones. The whole process is implemented using the MFT, to provide an effective and computationally efficient way of both separating the translation from the linear transformation and selecting model scale automatically.

4 Conclusions

Although much work remains to be done, the results shown here illustrate the usefulness of wavelet transforms in dealing with the symmetries associated with visual motions, both real and virtual. The orthonormal wavelet transform allows the iterated function compression scheme to be seen in a new light, which emphasises its connection with conventional image coding methods, while improving its performance significantly. The use of the MFT in video compression has illustrated the importance in general of having a representation which deals adequately with the problem of locality, as well as capturing the full $2-d$ affine group of symmetries. A novel 'featureless/feature-based' estimator using the MFT was briefly described. In both applications, results show the advantages of taking symmetries properly into account when selecting representations for image data compression.

This raises the perplexing question of why such symmetries, 'if they really exist', do not show up in image statistics, for in practice they never do. In the first place, to expect symmetries to reveal themselves in statistics such as the autocorrelation is as unreasonable as to expect a fair coin when tossed 100 times to yield precisely 50 heads and 50 tails. What can be said is that unless the number of heads deviates by an unlikely percentage from 50, the coin is fair: if you claim the world lacks a symmetry, then have adequate statistical grounds for that assertion. The visual world has an obvious geometrical structure [15], which ought to be reflected in the representations we use to explore it, even in such low level tasks as data compression, unless the statistics demand otherwise.

Acknowledgment

This work was supported by the UK EPSRC. Thanks to Andrew Calway and Stefan Kruger of the University of Bristol for supplying the motion estimates used in the sequence work.

Bibliography

1. I. Daubechies, I. (1992). *Ten Lectures on Wavelets.* SIAM Press, Penn.
2. Antonini, M., Barlaud, M., Mathieu, P. and Daubechies, I. (1992). Image coding using wavelet transform. *IEEE Trans. Image Process.*, **1**, 205-220.
3. Shapiro, J. (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Proc.*, **41**, 100-110.
4. Coifman, R.R. and Wickerhauser, M.V. (1992). Entropy-based algorithms for best basis selection. *IEEE Trans. Inform. Theory.* **38**, 713-718.
5. Jacquin, A.E. (1992). Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Trans. Image Proc.*. **1**, 18-30.
6. Barnsley, M.F. (1988). *Fractals Everywhere.* Academic Press, New York.

7. Jain, A.K. (1989). *Fundamentals of Digital Image Processing.* Academic Press. New York.

8. Lenz, R. (1990). *Group Theoretical Methods in Image Processing.* Springer-Verlag, Berlin.

9. Tekalp, A.M. (1995). *Digital Video Processing.* Prentice Hall, New Jersey.

10. T-I. Hsu, T-I. and Wilson, R. (1994). Texture analysis using a generalised wavelet transform. *Proc. IEEE ICIP-94.* **3**, Austin, 436-440.

11. R. Wilson, R., Calway, A.D. and Pearson, E.R.S. (1992). A generalized wavelet transform for Fourier analysis: the multiresolution Fourier transform and its application to image and audio signal analysis. *IEEE Trans. Inform. Theory.* **38**, 674-690.

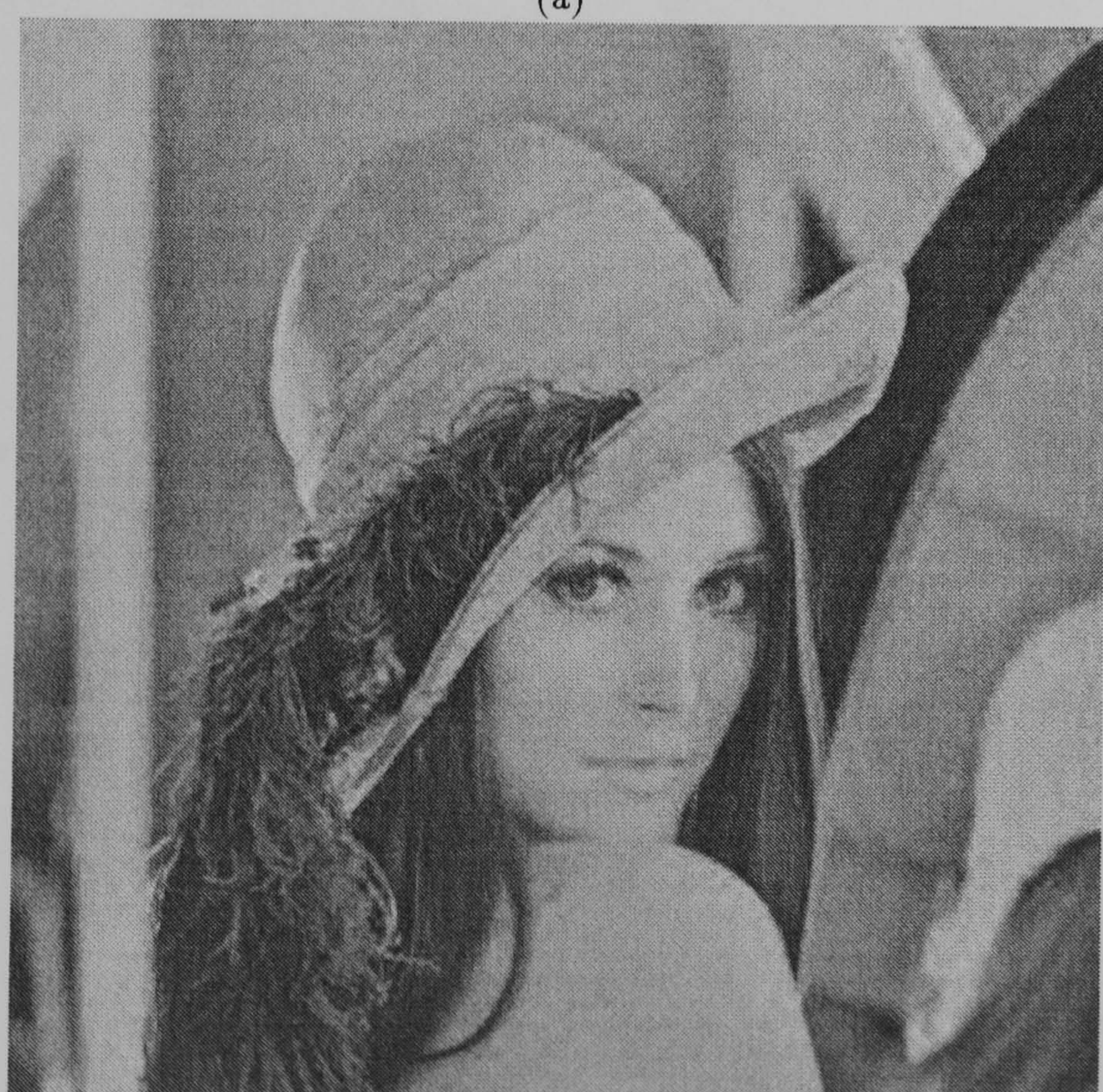
12. Antoine, J-P. (1996). Symmetry-adapted wavelet analysis. *Proc. IEEE ICIP-96.* **3**, Lausanne, 177-180.

13. S.A. Krüger, S.A. and A.D. Calway, A.D. (1996). A multiresolution frequency domain method for estimating affine motion parameters. *Proc. IEEE ICIP-96.* **1**, Lausanne, 113-116.

14. Davenport, W.B. and Root, W.L. (1987). *An Introduction to the Theory of Random Signals and Noise.* IEEE Press, New York, 312-350.

15. Gibson, J.J. (1974). *The perception of the visual world.* Greenwood Press, Conn.

(a)



(b)

Figure 1. (a)Wavelet fractal coding result using Daubechies 16-pt 'symmetric' orthonormal wavelets. Bit rate is 0.16bpp, output PSNR=31.49dB. (b) Bit rate 0.25bpp, PSNR=32.99dB.

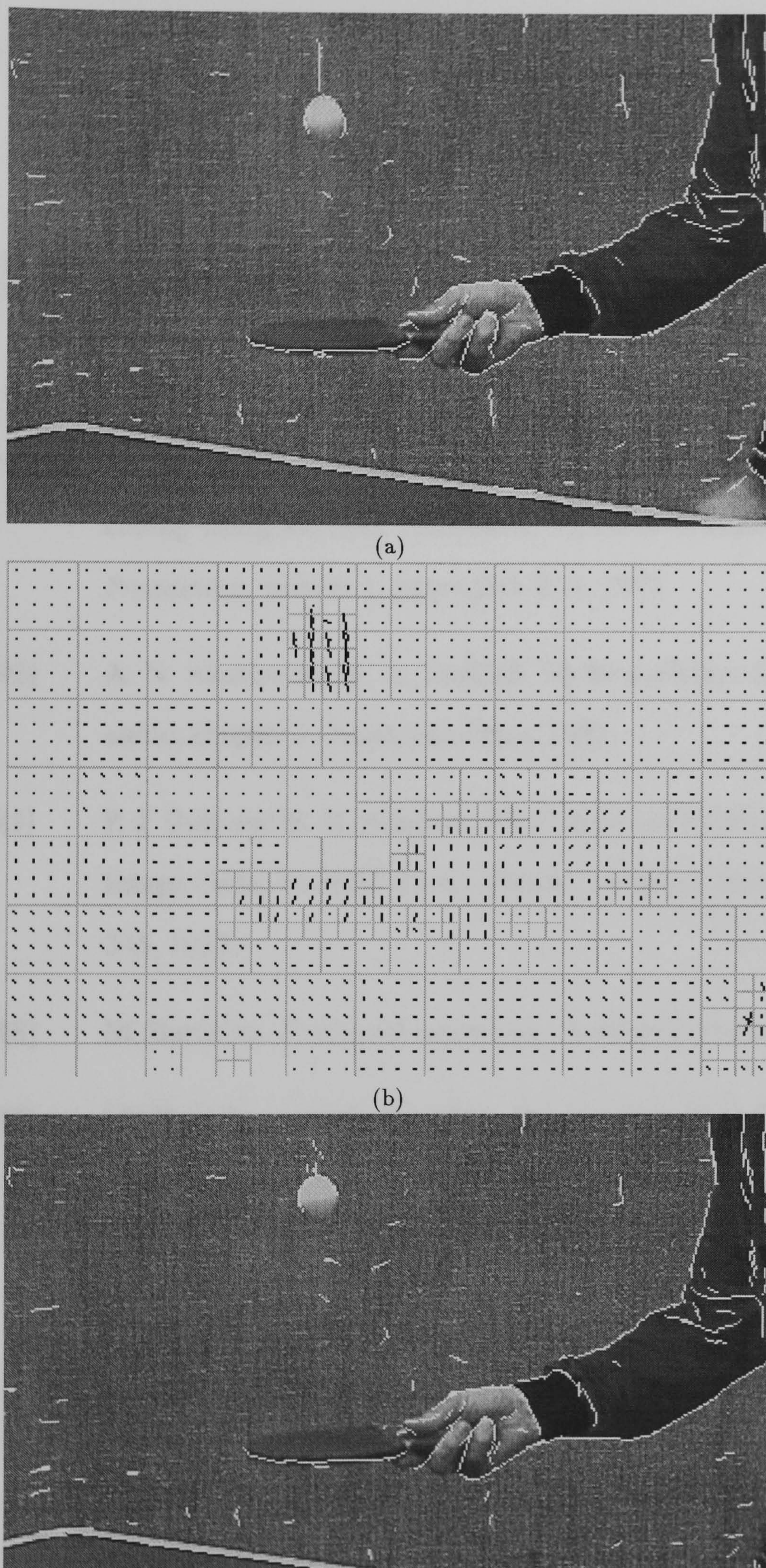


Figure 2. (a) Image from 'table tennis' sequence, with linear features overlaid. (b) Affine motion between frames, showing scale at which model was accepted. (c) Features from first frame, moved and overlaid on second frame.

Bibliography

[ABMD92] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using Wavelet Transform. In *IEEE Transactions on Image Processing*, volume 1, pages 205–219, 1992.

[AH92] A. N. Akansu and R. A. Haddad. *Multiresolution Signal Decomposition*, chapter 3. Academic Press, 1992.

[BA83] P. J. Burt and E. H. Adelson. The Laplacian Pyramid as a Compact Image Code. In *IEEE Trans. Comm.*, volume COM-31, pages 337–345, 1983.

[Bar88] M. Barnsley. *Fractals Everywhere.* Academic Press, 1988.

[BD85] M. Barnsley and S. Demko. Iterated Function Systems and the Global Construction of Fractals. In *Proc. Royal Society of London*, 1985.

[Ber68] T. Berger. Rate Distortion Theory for Sources with Abstract Alphabets and Memory. In *Inform. Contr.*, volume 13, pages 254–273, 1968.

[BH92] M. Barnsley and L. Hurd. *Fractal Image Compression*. AK Peters, 1992.

[BPL95] M. Balakrishnan, W. Pearlman, and L. Lu. Variable-Rate Tree-Structured Vector Quantizers. In *IEEE Transactions on Information Theory*, volume 41, pages 917–930, 1995.

[Bra86] Ronald N. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill, 1986.

[BV94] K. Barthel and T. Voyé. Adaptive Fractal Image Coding in the Frequency Domain. In *Proceedings of International Workshop on Image Processing : Theory, Methodology, Systems and Applications*, 1994.

[Cho96] C-H. Chou. Low Bit Rate 3-D Subband Video Coding Based On The Allocation of Just Noticable Distortion. In *Proceedings of International Conference on Image Processing*, volume I, pages 637–640, 1996.

[Chu92] C. Chui. *An Introduction to Wavelets*, chapter 1. Academic Press, 1992.

[Cla85] R. J. Clarke. *Transform Coding of Images*. Academic Press, 1985.

[Cla95] R. J. Clarke. *Digital Compression of Still Images and Video*. Academic Press, 1995.

[CMW] R. Coifman, Y. Meyer, and V. Wickerhauser. Adapted Waveform Analysis,
Wavelet-Packets and Applications. Technical report, Department of Mathematics, Yale University. Retrieved from World Wide Web at ftp://pascal.math.yale.edu/pub/wavelets/adaptwave1.tex.

[CRA96] L. Corte-Real and A. P. Alves. A Very Low Bit Rate Video Coder Based on Vector Quantization. In *IEEE Transactions on Image Processing*, volume 5, pages 263–273, 1996.

[Dau88] I. Daubechies. Orthonormal Bases of Compactly Supported Wavelets. In *Communications on Pure and Applied Mathematics*, volume XLI, pages 910–996, 1988.

[Dau92] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, Philadelphia, PA, 1992. Notes from the 1990 CBMS-NSF Conference on Wavelets and Applications at Lowell, MA.

[Edi91] Special Edition. Digital Multimedia Systems. In *Communications of the ACM*, volume 34, 1991.

[Fis95] Y. Fisher. *Fractal Image Compression : Theory and Application*. Springer-Verlag, 1995.

[FJF77] J. Friedman, J.Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. In *ACM Trans. Math. Software 3,3*, pages 209–226, 1977.

[FM95] Y. Fisher and S. Menlove. Fractal Encoding with HV Partitions. In *Fractal Image Compression : Theory and Applications*, pages 119–135, 1995.

[GAH94] M. Gharavi-Alkhansari and T. Huang. Fractal-Based Techniques for a Generalized Image Coding Method. In *Proc. IEEE ICASSP*, pages 122–126, 1994.

[GLGO94] R. A. Gopinath, M. Lang, H. Guo, and J. E. Odegard. Wavelet-Based Post-Processing of Low Bit Rate Transform Coded Images. Technical Report CML TR94-15, Rice University, 1994.

[Gra67] D. N. Graham. Image Transmission by Two-Dimensional Contour Coding. In *Proc. IEEE*, volume 55, pages 336–346, 1967.

[GW92] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[HMS96] R. Hamzaoui, M. Muller, and D. Saupe. VQ-Enhanced Fractal Image Compression. In *Proceedings of International Conference on Image Processing*, volume I, pages 153–156, 1996.

[Jac90] A. E. Jacquin. A Novel Fractal Block-Coding Technique for Digital Images. In *Proc. ICASSP '90*, pages 18–30, 1990.

[Jai89] A. K. Jain. *Fundamentals of Digital Image Processing*, chapter 11. Prentice Hall, 1989.

[JFB91] E. Jacobs, Y. Fisher, and R. Boss. Iterated Transform Image Compression. Technical Report 1408, Naval Ocean Systems Centre, April 1991.

[JFB92] E. Jacobs, Y. Fisher, and R. Boss. Image Compression : A study of the iterated transform method. *Signal Processing*, 29:251–263, 1992.

[KMK95] H. Krupnik, D. Malah, and E. Karnin. Fractal Representation of Images via the Discrete Wavelet Transform. In *IEEE 18th Conv. of EE*, 1995.

[Lan84] G. Langdon. An Introduction to Arithmetic Coding. In *IBM Journal of Research and Development*, volume 28, pages 135–149, 1984.

[LBG80] Y. Linde, A. Buzo, and R Gray. An Algorithm for Vector Quantizer Design. In *IEEE Transactions on Communications*, volume 28, pages 84–95, 1980.

[LeG92] D. J. LeGall. The MPEG video compression algorithm. In *Signal Proc.: Image Comm.*, volume 4, pages 129–140, 1992.

[LLK96] J. Li, J. Li, and C.-C. Jay Kuo. An Embedded DCT Approach to Progressive Image Compression. In *Proceedings of International Conference on Image Processing*, volume I, pages 201–204, 1996.

[LW96] I. Levy and R. Wilson. Predictive Wavelet Transform Coding : Unifying Fractal and Transform coding. In *Proceedings PCS96*, pages 347–352, Melbourne, 1996.

[Mal89] S. Mallat. Multiresolution approximation and wavelet orthonormal bases of $l^2(\Re)$. In *Trans. Amer. Math Soc.*, volume 315, pages 69–87, 1989.

[MD92] D. Monro and F. Dudbridge. Fractal Approximation of image Blocks. In *Proc. IEEE ICASSP*, pages 485–488, 1992.

[MDW90] D. Monro, F. Dudbridge, and A. Wilson. Deterministic rendering of self-affine fractals. In *IEE Colloquium on Fractal Techniques in Image Processing*, 1990.

[Mon93] D. Monro. Class of Fractal Transform. In *Electronics Letters*, volume 29, pages 362–363, 1993.

[Mou69] F. W. Mounts. A Video Encoding System With Conditional Picture-Element Replensihment. In *Bell Systems Technical Journal*, pages 2545–2554, September 1969.

[MW94] D. Monro and S. Woolley. Fractal Image Compression Without Searching. In *Proc. IEEE ICASSP*, pages 557–560, 1994.

[MWN93] D. Monro, D. Wilson, and J. Nicholls. High Speed Image Coding with the Bath Fractal Transform. In *Multimedia*, 1993.

[Ner69] E. D. Nering. *Linear Algebra and Matrix Theory*, chapter 3. Wiley, 1969.

[PC94] L-M. Po and C-K. Chan. Adaptive Dimensionality Reduction Techniques for Tree-structured Vector Quantization. In *IEEE Transactions on Communications*, volume 42, 1994.

[Pea95] D. E. Pearson. Developments in Model-Based Video Coding. In *Proceedings of the IEEE*, volume 83, pages 892–906, 1995.

[Pin69] J. T. Pinkston. An Application of Rate-Distortion Theory to a Converse to the Coding Theorem. In *IEEE Trans. Inform. Theory*, volume 15, pages 66–71, 1969.

[PJF95] C. I. Podilchuk, N. S. Jayant, and N. Farvardin. Three Dimensional Subband Coding of Video. In *IEEE Transactions on Image Processing*, volume 4, pages 125–139, 1995.

[PM92] W. Pennebaker and J. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, 1992.

[PM96] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing, Principles, Algorithms and Applications.* Prentice Hall, 1996.

[RA86] B. Ramamurthi and A.Gersho. Classified Vector Quantization of Images. *IEEE Trans. Commun.*, 34:1105–1115, Nov 1986.

[RC95] R. Rinaldo and G. Calvagno. Image Coding by Block Prediction of Multiresolution Subimages. Technical report, Universitá di Padova, 1995.

[RV93] K. Ramchandran and M. Vetterli. Best Wavelet Packet Bases in a Rate-Distortion Sense. In *IEEE Transactions on Image Processing*, volume 2, pages 160–175, 1993.

[Sau95] D. Saupe. Accelerating Fractal Image Compression by Multi-Dimensional Nearest Neighbour Search. In *Proceedings DCC'95 Data Compression Conference*, 1995.

[Sha48] C. E. Shannon. A Mathematical Theory of Communication. In *Bell System Technical Journal*, volume 27, pages 379–423; 623–656, 1948.

[Sha59] C. E. Shannon. Coding Theorems for a Discrete Source with a Fidelity Criterion. In *IRE Nat. Conv. Rec.*, volume 4, pages 142–163, 1959.

[Sha93] J. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. In *IEEE Transactions on Signal Processing*, volume 41, pages 3445–3462, 1993.

[SMP97] M. K. Steliaros, G. R. Martin, and R. A. Packwood. Locally-Accurate Motion Estimation for Object-Based Video Coding. In *SPIE*, volume 3309, pages 306–316, 1997.

[SR96] D. Saupe and M. Ruhl. Evolutionary Fractal Image Compression. In *Proceedings of International Conference on Image Processing*, volume I, pages 129–132, 1996.

[Tek95] A. M. Tekalp. *Digital Video Processing*, chapter 23. Prentice Hall, 1995.

[TMB89] N. Treil, S. Mallat, and R. Bajcsy. Image Wavelet Decomposition and Applications. Technical Report MS-CIS-89-22, University of Pennsylvania, Department of Computer and Information Science, April 1989.

[Tod89] M. P. Todd. *Image Data Compression Based On a Multiresolution Signal Model*. PhD Thesis, Univeristy of Warwick, 1989.

[Wat88] R. Watt. *Visual Processing : Computational, psychophysical and cognitive research*, chapter 6. Lawrence Erlbaum Associates Ltd, 1988.

[Wil87] R. Wilson. Finite Prolate Spheroidal Sequences and Their Applications I: Generation and Properties. In *IEEE Trans.*, volume PAMI-9, 1987.

[Wil91] R. Williams. *Adaptive Data Compression*, chapter 1. Kluwer Academic Publishers, 1991.

[Wil97] R. Wilson. Private Communication, 1997. via e-mail. Wilson supplied the filter coefficients used in Chapter 4.

[WNC87] I. Witten, R. Neal, and J. Cleary. Arithmetic Coding for Data Compression. In *Communications of the ACM*, volume 30, pages 520–540, 1987.

[WS88] R. Wilson and M. Spann. *Image Segmentation and Uncertainty*. Wiley, 1988.

[ZM90] S. Zhong and S. Mallat. Compact Image Representation from Multiscale Edges. In *3rd International Conference on Computer Vision*, pages 522–525, 1990.